Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

Tuomas Yrjänäinen

HTML5 Canvas -ohjelmointikirjastot

Diplomityö
Espoo, 31. heinäkuuta 2016

Valvoja: Professori Petri Vuorimaa
Ohjaaja: Diplomi-insinööri Aleksi Aalto

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Tuomas Yrjänäinen

Title:
HTML5 Canvas software libraries

Date: July 31, 2016 **Pages:** vii + 61

Major: Media Technology **Code:** T-111

Supervisor: Professor Petri Vuorimaa

Advisor: Aleksi Aalto M.Sc. (Tech.)

The development of websites is heading towards plugin-free architecture. HTML5 specification introduced a new Canvas element that includes graphical properties which could previously only be achived by using browser add-ons.

This thesis examines software libraries that are geared towards development of HTML5 Canvas applications. The aim of this study is to identify a software library that is efficient, suitable for commercial use and meets the other requirements presented in this study.

This study analysed 42 software libraries, which were published using open source license that enables commercial use. Seven libraries met the requirements of this study regarding technical features and qualifications that were required from the development project of the software library. These seven libraries were selected for performance and memory usage testing.

The performance tests were done using sample programs that drew vector and bitmap graphics on HTML5 Canvas. The sample programs were measured for redraw speed of the graphics and memory usage. These tests were carried out on desktop and mobile environments with various number of graphical elements.

In conclusion results of the study showed that there are clear differences between HTML5 Canvas libraries regarding performance and memory usage. Out of all libraries Easel.js and Pixi.js libraries were selected as the best HTML5 Canvas libraries.

Keywords: HTML5, Canvas, JavaScript, Software Library, performance, memory usage

Language: Finnish

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Tuomas Yrjänäinen		
Työn nimi:	HTML5 Canvas -ohjelmointikirjastot		
Päiväys:	31. heinäkuuta 2016	**Sivumäärä:**	vii + 61
Pääaine:	Mediatekniikka	**Koodi:**	T-111
Valvoja:	Professori Petri Vuorimaa		
Ohjaaja:	Diplomi-insinööri Aleksi Aalto		

Verkkosivujen tekninen kehitys on siirtymässä vahvasti ilman selainliitännäisiä toimiviin sivustoihin. HTML5-standardin mukana kehitetty Canvas-elementti tarjoaa ne graafiset ominaisuudet, jotka oli aiemmin mahdollista toteuttaa vain selainliitännäisten avulla.

Tässä tutkimuksessa tarkastellaan HTML5 Canvas -elementin ohjelmointiin tarkoitettuja ohjelmointikirjastoja. Tavoitteena on löytää tehokas, kaupalliseen käyttöön soveltuva kirjasto, joka täyttää tutkimuksessa asetetut vaatimukset.

Tutkimuksessa tarkasteltiin 42 ohjelmointikirjastoa, jotka on julkaistu kaupallisen käytön sallivalla avoimen lähdekoodin lisenssillä. Kirjastoista seitsemän täytti tutkimuksessa asetetut, teknisiä ominaisuuksia ja kehitysprojektin laatua koskevat vaatimukset. Näille kirjastoille tehtiin suoritustehokkuutta ja muistinkäyttöä mittaavat testit.

Teknisessä testauksessa ohjelmointikirjastoilla koostettiin vektori- ja bittikarttagrafiikkaa piirtävät vertailuohjelmat. Testiohjelmilla mitattiin ruudunpäivitysnopeus sekä varatun muistin määrä. Testit suoritettiin sekä pöytätietokoneella että mobiiliympäristössä varioiden ruudulle piirrettävän grafiikan määrää.

Tutkimuksessa saadut tulokset osoittavat, että HTML5 Canvas -ohjelmistokirjastoilla on selviä eroja suoritustehokkuudessa sekä muistinkäytössä. Tutkituista ohjelmointikirjastoista parhaiksi valikoituivat Easel.js ja Pixi.js.

Asiasanat:	HTML5, Canvas, JavaScript, ohjelmointikirjasto, suorituskyky, muistin käyttö
Kieli:	suomi

Alkusanat

Kiitän professori Petri Vuorimaata työn valvonnasta ja Aleksi Aaltoa työn ohjaamisesta. Suuret kiitokset työtä eteenpäin vieneistä ideoista ja kaikesta avusta.

Espoo, 31. heinäkuuta 2016

Tuomas Yrjänäinen

Lyhenteitä ja käsitteitä

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
FPS	Frames Per Second
GPL	GNU General Public License
HTML	HyperText Markup Language
HTTP	HyperText Tranfer Protocol
IETF	Internet Engineering Task Force
JDK	Java Development Kit
LGPL	GNU Lesser General Public License
MBR	Minimum Bounding Rectangle
SGML	Standard Generalized Mark-up Language
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
WebGL	Web Graphics Library
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

Sisältö

Luku 1

Johdanto

Internetselaimessa HTML-tekniikalla (HyperText Markup Language) toimivat käyttöliittymät ja sovellukset ovat muodostuneet yleiseksi standardiksi ohjelmistoalalla. Sovellukset, joiden käyttöliittymät eivät vaadi monipuolista vapaata graafista esitystä, ovat nykyään pääsääntöisesti toteutettu HTML-tekniikalla. HTML-dokumentin rakenteen rajoitukset ovat kuitenkin johtaneet siihen, että HTML-standardiin kuulumattomien ominaisuuksien puutteita on vuosien saatossa kierretty erilaisilla selainliitännäisillä. Esimerkiksi Java- ja Flash-selainliitännäiset ovat tarjonneet keinon laajentaa HTML-dokumentin graafisia mahdollisuuksia. Ongelmana näiden liitännäisten käytössä on ollut avoimien standardien puute, riippuvuus yksittäisestä kaupallisesta toimijasta, tietoturvaongelmat sekä liitännäisten toimimattomuus mobiililaitteissa.

Lokakuussa 2014 julkaistiin virallinen HTML5-standardi, jonka yhteydessä määriteltiin HTML-dokumentin Canvas-elementti. Sen avulla on mahdollista toteuttaa toiminnallisuuksia, jotka aiemmin jouduttiin toteuttamaan erilaisilla selainliitännäisillä[1]. Canvas-elementti on piirtoalue, jonka sisälle piirrettävää grafiikkaa voidaan kontrolloida pikselitasolla. Se eroaa perinteisistä HTML-dokumentin elementeistä, jotka selain piirtää ruudulle vakiomäärittelyjen mukaisesti.

Canvas-elementin käyttö ei kuitenkaan ole ongelmatonta ohjelmistokehityksessä: Elementin API-kutsut (Application Programming Interface) sisältävät ainoastaan suppean joukon matalan tason graafisia piirtokomentoja. Tehokkaan ohjelmistokehityksen kannalta tärkeitä korkeamman tason toiminnallisuuksia, kuten graafisten elementtien törmäyksen tarkistusta, ei ole liitetty mukaan standardiin. Esimerkiksi Flash-liitännäisen käyttämä ActionScript 3 -ohjelmointikieli sisältää vakiona suuren joukon käyttöliittymäohjelmointia nopeuttavia tekniikoita[2]. Jos Canvas-elementin avulla kehitetään esimerkiksi käyttöliittymää sovellukseen, joudutaan kir-

joittamaan paljon ohjelmakoodia jo pelkästään perustoiminnallisuuksien toteuttamiseksi. Tästä syystä Canvas-elementin ohjelmoinnissa on tarpeen käyttää ohjelmointikirjastoja. Niiden avulla voidaan saavuttaa sama kustannustehokkuus Canvas-elementin ohjelmoinnissa kuin HTML-dokumenteissa käytettävien selainliitännäisten kanssa.

Canvas-elementille ei ole olemassa yhtä standardoitua ohjelmointikirjastoa, vaan sille on kehitetty useita, hieman eri osa-alueisiin painottuneita kirjastoja. Niitä on julkaistu sekä avoimella lähdekoodilla sekä kaupallisina tuotteina. Osa niistä on keskittynyt yleiskäyttöisyyden parantamiseen laajentamalla Canvas-elementin piirto-ominaisuuksia, kun taas osa palvelee erityisesti esimerkiksi pelikehityksen tarpeita.

1.1 Tutkimuskysymykset

Tämän tutkimuksen tavoitteena on kartoittaa hyödyllisimmät ohjelmointikirjastot HTML5 Canvas -elementillä toteutettavien sisältöjen tekemiseen. Tutkimuksessa selvitetään eri kirjastojen tukemat tekniset ominaisuudet ja tarkastellaan miten aktiivinen kehitysyhteisö ja käyttäjäkunta ohjelmistokirjastolla on. Lisäksi tutkimuksessa vertaillaan teknisesti ohjelmointikirjastojen piirtotehokkuutta ja muistinkäyttöä. Tavoitteena on löytää tehokkuudeltaan hyvä HTML5 Canvas -ohjelmointikirjasto, joka sopii yleiskäyttöisesti Canvas-elementin ohjelmointiin ja jonka kehitysyhteisö on aktiivisesti kehittämässä kirjastoa. Tämän tavoitteen saavuttamiseksi tutkimuksessa on asetettu seuraavat tutkimuskysymykset:

- Mitkä tekniset ominaisuudet ohjelmointikirjastot täyttävät?

Tutkimuksessa tarkastellaan ohjelmointikirjastojen teknisiä ominaisuuksia. Tarkoituksena on todentaa, että kirjastot sisältävät ne ominaisuudet, jotka ovat keskeisiä graafisen- ja erityisesti käyttöliittymäohjelmoinnin kannalta. Tarkoituksena on myös arvioida ohjelmointikirjastojen ominaisuuksien laajuutta ja karsia teknisesti testattavista ohjelmointikirjastoista ne, jotka eivät sisällä riittävän laajasti tarvittavia toiminnallisuuksia.

- Miten aktiivinen kehitysyhteisö ohjelmointikirjastolla on?

Tutkimuksessa selvitetään ohjelmistokirjaston kehitysyhteisön aktiivisuutta. Kehitysyhteisön toimintaa tutkitaan tarkastelemalla statistiikkaa kirjaston versionhallinnasta. Tarkoituksena on löytää kirjastot, joiden kehitystyö on aktiivista, ja valita niistä lupaavimmat tekniseen testaukseen.

- Mitä eroja on tutkittavien ohjelmointikirjastojen suorituskyvyssä?

Suorituskyvystä tutkitaan ensisijaisesti kirjaston graafista suorituskykyä. Sitä arvioidaan mittaamalla eri ohjelmistokirjastoilla tehdyn referenssiohjelman piirtonopeutta. Piirtonopeus mitataan laskemalla ruudunpäivitysten määrä sekunnissa erilaisissa testeissä. Toissijaisena suorituskyvyn mittarina tutkitaan referenssiohjelman ajonaikaista JavaScript-suorituksen muistinkäyttöä. Muistinkäytöllä on merkitystä etenkin mobiilipäätelaittella, joissa muistia on rajoitetummin käytössä kuin pöytäkoneissa.

1.2 Diplomityön rakenne

Tutkimus etenee seuraavasti: Luvuissa 2–3 taustoitetaan tutkimusta esittelemällä HTML-tekniikkaa ja ohjelmointikirjastoja. Luvussa 4 käydään tarkemmin läpi testi- ja tutkimusmenetelmät ohjelmointikirjastojen vertailua varten. Luvussa 5 suoritetaan esikarsinta tutkittaville ohjelmointikirjastoille. Luvussa 6 tarkastellaan lähemmin ohjelmointikirjastojen ominaisuuksia. Luvussa 7 esitellään tehokkuustestaukseen valittujen ohjelmointikirjastojen tulokset. Luvussa 8 analysoidaan testattujen kirjastojen tuloksia. Luvussa 9 esitetään yhteenveto tutkimuksen tuloksista.

Luku 2

HTML-tekniikka

2.1 HTML-tekniikan kehitys

HTML-tekniikan kehitys alkoi Cernissä 1989, kun Tim Berners-Lee ryhtyi kehittämään dokumenttien kirjoittamiseen tarkoitettua hypertekstijärjestelmää. Tavoitteena oli luoda järjestelmä, jonka avulla tieteellisiä artikkeleja olisi mahdollista kirjoittaa hajautetusti. Useiden kirjoittajien tekstejä ei yhdistettäisi yhdeksi isoksi dokumentiksi lataamalla ne samalle tietokoneelle, vaan yhdistäminen tapahtuisi linkittämällä eri kirjoittajien tekstit hyperlinkkien avulla. Päämääränä oli kehittää hypertekstijärjestelmälle formaatti, joka olisi laite- ja käyttöjärjestelmäriippumaton, eikä tarvitsisi toimiakseen muita aiemmin käytettyjä dokumenttiformaatteja, kuten Microsoft Wordia tai Latexia. [3] Berners-Lee julkaisi ensimmäisen kuvauksen HTML-kielestä vuonna 1991 [4].

HTML-kuvauskielen kehitys pohjautui vahvasti SGML-kuvauskieleen (Standard General Markup Language), joka oli jo käytössä maailmanlaajuisesti. HTML-dokumentin määrittely otti useita elementtejä suoraan SGML-kielestä. Esimerkiksi tekstin ja eri tasoisten otsikoiden kuvaamiseen tarkoitetut <P>- sekä <H1>–<H6>-elementit oli jo toteutettu SGML-kielessä. Yksi tärkeimmistä uusista määritelmistä HTML-kielessä oli <A>-ankkurielementti, jolla voitiin esittää hypertekstilinkkejä. [3, 5]

HTML-dokumentin muoto suunniteltiin alun perin palvelemaan lähinnä staattisen sisällön esittämistä. Mallia HTML-dokumenttiin tarvittavista elementeistä otettiin lehdistä sekä muista painetuista medioista. HTML-dokumentin haluttiin sisältävän samanlaisia elementtejä kuin painetussa teksteissä, jotta samoja sisältöjä voitaisiin julkaista sekä painettuna että HTML-muodossa. [3]

HTML-dokumentin ohella Berners-Lee kehitti myös ensimmäisen HTML-

dokumenttien jakamiseen ja näyttämiseen tarkoitetut palvelin- ja selainohjelmistot vuonna 1991 [6]. Berners-Leen antoi kehittämänsä selaimen nimeksi WorldWideWeb, joka viittasi HTML-dokumenttijärjestelmän nimeen World Wide Web alkuperäisessä projektisuunnitelmassa [7, 8]. Samalla Berners-Lee kehitti ensimmäisen version HTML-dokumenttien lataamiseen tarkoitetusta HTTP-protokollasta (HyperText Transfer Protocol) [9]. Myöhemmin HTTP-protokollan kehitys siirtyi IETF-organisaation (Internet Engineering Task Force) hallintaan ja HTTP-protokollan kehitykseen liittyi mukaan myös muita henkilöitä [10].

1990-luvun alussa HTML-kielen rakenne ja standardointimenettelyt eivät olleet vielä vakiintuneet. Muutamat ensimmäiset vuodet HTML-dokumentilla ei ollut selkeää standardia. Berners-Leen vuonna 1991 www-talk keskusteluryhmään kirjoittama viesti sisältää lyhyen kuvauksen HTML-elementeistä ja linkin ensimmäisten HTML-elementtien määrittelyyn [4, 11]. Virallisemman standardin esitti IETF vuonna 1993 julkaisemassaan HTML-määrittelyssä [12]. Seuraava standardi HTML 2.0 julkaistiin vuonna 1995, kun IETF:n aiempi luonnos HTML-kielen määrittelystä vanhentui [13]. Vuonna 1997 julkaistun HTML 3.2:n myötä HTML-kielen standardien esittäminen siirtyi W3C-organisaation tehtäväksi [14]. W3C[1] on Tim Berners-Leen vuonna 1994 perustama organisaatio, jonka tarkoitus on standardisoida ja kehittää HTML-tekniikkaa. Vielä vuonna 1997 W3C julkaisi myös HTML 4.0 -standardin, josta muokattu HTML 4.01 -standardi julkaistiin vuonna 1999 [15, 16]. HTML 4.01 -standardin jälkeen seuraava virallinen HTML-standardi HTML5 julkaistiin vasta 2014 [1].

HTML 4.01 -standardin julkaisun jälkeen W3C-organisaation näkemys HTML-kielen kehityksestä oli siirtyä XML-pohjaiseen (Extensible Markup Language) XHTML-kieleen (Extensible Hypertext Markup Language). Ajatuksena oli, että HTML-dokumenttiin voitaisiin lisätä uusia elementtejä ilman erillistä määrittelyä. Toinen W3C:n asettama tavoite oli luoda tarkempi virheiden käsittely verkkosivujen syntaksille. XHTML-dokumenttien haluttiin noudattavan tarkemmin määriteltyä muotoa, jonka ajateltiin johtavan yhtenäisempään HTML-koodiin. Todellisuudessa vaatimus täydellisestä XHTML-syntaksista olisi tuottanut tilanteita, joissa pienikin virhe ohjelmakoodissa olisi aiheuttanut verkkosivun toimimattomuuden. [17]

2000-luvun alussa selainvalmistajien käsitykset HTML-kielen tulevaisuudesta alkoivat erota W3C:n näkemyksestä. W3C:n päämääränä olivat staattiset ja virheettömät XHTML-dokumentit; selainvalmistajien mielestä HTML-tekniikan pitäisi fokusoida enemmän selaimen kautta käytettäviin dynaamisiin applikaatioihin. W3C:n suuntasi kehityksensä XHTML 2.0:n

[1] W3C http://w3.org

määrittelyyn. Se ei olisi ollut taaksepäin yhteensopiva eikä olisi sisältänyt selainvalmistajien toivomia ominaisuuksia. Vuonna 2004 W3C:n rinnalle syntyikin Applen, Mozillan ja Operan aloittamana selainvalmistajien oma yhteenliittymä WHATWG[2] (Web Hypertext Application Technology Working Group), joka ryhtyi kehittämään omaa versiotaan HTML-kielestä. Koska XHTML-tekniikka ei saavuttanut suosiota, W3C ryhtyi lopulta kehittämään HTML5-standardia WHATWG-ryhmän kanssa. XHTML 2.0 -määrittely hylättiin lopullisesti vuonna 2009. [17, 18]

Vuonna 2011 WHATWG-ryhmän näkemys erkani kuitenkin W3C:n HTML5-standardin luonnoksesta. W3C:n tavoitteena oli määritellä joukko ominaisuuksia, joista muodostuisi HTML5-määrittely. WHATWG taas keskittyi luomaan HTML-määrittelyä, jota ei olisi rajattu tarkasti versionumeroon, vaan se päivittyisi aina uusien tekniikoiden kehittyessä. [19, 20]

2.1.1 HTML-standardoinnin tulevaisuus

Virallisen HTML5-standardin julkaisun jälkeen siitä erotettiin omaksi standardoitavaksi moduulikseen erilaiset toiminnallisuudet, kuten esimerkiksi Canvas-elementti [21]. Menettelyn tarkoituksena on nopeuttaa standardoinnin etenemistä siten, että yhden osa-alueen hitaammin etenevä määrittelytyö ei hidasta koko määrittelytyön etenemistä. Omiksi moduuleikseen jaetut ominaisuudet ovat kuitenkin edelleen osa HTML-standardia.

W3C:n on tarkoitus julkaista HTML5.1:n virallinen määrittely vuoden 2016 loppupuolella. Muutokset standardiin eivät tule olemaan kovin suuria: HTML5.1:ssä kehitetään muutamia uusia elementtejä ja nimetään lisää teknisesti vanhentuneita HTML-elementtejä, joita ei enää suositella käytettäväksi. [22, 23]

2.2 HTML-dokumentin rakenne ja tekniikka

HTML on kuvauskieli, jolla voidaan kuvata web-sivun rakenne. HTML-dokumentti rakentuu sisäkkäisistä elementeistä, jotka määrittelevät HTML-dokumentin sisällön rakenteen (esimerkki 2.1). Selaimen ladatessa HTML-dokumentin siitä muodostetaan selaimen käyttämä DOM-malli (Document Object Model). [24]

DOM-malli on ohjelmointirajapinta, joka kuvaa kaikki HTML-dokumentin elementit ja niiden suhteet toisiinsa. DOM-malli luo HTML-elementtien rakenteista puumallisen esityksen, joka vastaa HTML-dokumentin rakennetta.

[2]WHATWG https://whatwg.org

Tämän rajapinnan kautta on mahdollista käsitellä verkkosivun sisältöä dynaamisesti JavaScript-kielellä. [25]

Nykyaikaisessa HTML-dokumentissa ulkoasun määrittely on usein erotettu sisällöstä. HTML-dokumenttien ulkoasun kuvaamiseen käytetään CSS-kieltä (Cascading Style Sheets). CSS-kielellä voidaan määritellä HTML-dokumentissa esitettyjen elementtien graafiset parametrit (esimerkki 2.2). [26]

```
<!doctype html>
<html>
  <head>
    <title>Esimerkkisivu</title>
    <style src="tyyli.css"></style>
    <script src="kirjasto.js"></script>
  </head>
  <body>
    <h1>Esimerkkiotsikko</h1>
    <p class="description">Esimerkkitekstikappale</p>
    <canvas id="testCanvas"></canvas>
  </body>
</html>
```

Esimerkki 2.1: Yksinkertainen HTML5-dokumentti.

```
h1 {
  color: #000;
  font-size:20px;
}

#testCanvas {
  height:900px;
  width:1200px;
}

.description {
  font-size:10px;
}
```

Esimerkki 2.2: Yksinkertainen CSS-määrittely.

2.2.1 JavaScript

JavaScript on alun perin Netscape-selaimessa vuonna 1995 julkaistu ohjelmointikieli, jota käytetään verkkosivujen dynaamisten toiminnallisuuksien ohjelmointiin. Tarkoituksena oli luoda kieli, joka olisi suosittua Java-kieltä yksinkertaisempi ja samalla helpompi ohjelmoitava. [27]

JavaScript-kielellä on mahdollista muokata selaimessa verkkosivujen DOM-puun sisältämiä elementtejä ja käyttää elementtien ohjelmointirajapintoja. Näihin kuuluvat mm. Canvas-elementin tarjoamat JavaScript API -kutsut [21]. JavaScript on dynaamisesti tyypitetty tulkattava kieli, jota suoritetaan käyttämällä selaimien JavaScript-tulkkia. Tulkattavana kielenä sen suoritusnopeus riippuu paljon selaimen JavaScript-tulkin implementaatiosta. [28] JavaScript-kielen suorituskyky arvioitiin vuonna 2011 50 kertaa hitaammaksi kuin C-kielellä toteutetun vertailuohjelman [29].

Nykyaikaisten selainten käyttökelpoisuus riippuu paljon siitä, miten tehokkaasti selain pystyy suorittamaan HTML-sivun toiminnallisuuksia ohjaavaa JavaScript-kieltä. Tästä johtuen suurimmat selainvalmistajat ovat tehneet paljon kehitystyötä selaimien JavaScript-tulkkien tehostamiseksi. Esimerkiksi Google on käyttänyt paljon resursseja Chrome-selaimessa käytettävän V8 JavaScript-tulkin kehittämiseen, samoin Mozilla Foundation Firefox-selaimen Spidermonkey JavaScript-tulkkin kehitystyöhön. [30–32]

2.3 Piirtoalueet HTML-dokumentissa

Alkuperäinen HTML-dokumentti oli tarkoitettu lähinnä staattisen tiedon esittämiseen. HTML-dokumentissa oli mahdollista esittää kuvia ja tekstiä, mutta kaikki monipuolisempi sisältö puuttui alkuvaiheessa kokonaan [8]. Tarve dynaamisen sisällön esittämiseen verkkosivuilla lisääntyi HTML-sivujen yleistyessä. Ratkaisuksi kehitettiin HTML-dokumentin sisään liitettäviä sovelmia (applet). Sovelmat ovat erillisiä ohjelmia, joita voidaan suorittaa HTML-dokumentin sisällä. Tällä tavoin HTML-dokumentin sisälle voitiin liittää monimutkaisempaa grafiikkaa, videota ja ääntä esittäviä sisältöjä.

1990-luvulla yksi suosituimmista sovelmatekniikoista olivat Java-sovelmat. Verkkosivun sisään voitiin liittää erillinen Java-ohjelma, jolla oli mahdollista luoda monipuolisia graafisia sisältöjä kuten pelejä ja simulaatioita. Java-tekniikka levisi alun perin laajalle, koska se oli käyttöjärjestelmäriippumaton. [33] Jotta Java-sovelmaa voidaan käyttää, koneelle pitää olla kuitenkin asennettuna Java-liittännäinen, mikä luo uuden teknisen riippuvuuden [34]. Java-sovelmien suosio laski tietoturvaongelmien ja muiden selainliittännäisten suosion nousun takia. Java-selainliitännästä ylläpitävä Oracle on vuoden 2016 alussa päättänyt lopettaa Java-selainliitännäisen kehittämisen JDK 9 (Java Development Kit) julkaisun yhteydessä [35].

Java-sovelmien jälkeen suositummaksi selainlaajennustekniikaksi nousi Adobe Flash -selainlaajennus. Se mahdollisti ennen HTML5-tekniikkaa äänen, videon ja monimutkaisen grafiikan esittämisen selaimessa. Flash-

selainliitännäinen nousi de facto -standardiksi selaimissa osaksi videotoistotekniikan ansiosta. Suuri osa videoiden esittämiseen keskittyneistä sivustoista, kuten Youtube[3], aloitti käyttämällä Flash-tekniikkaa videoiden toistossa. Flash-videotekniikan kehittyminen yhdessä internetliittymien nopeutumisen kanssa toivat videot osaksi normaaleja verkkosivuja. [36]

Flash-selainlaajennuksen käyttäminen on ollut ongelmallista muissa kuin Windows-käyttöjärjestelmällä varustetuissa tietokoneissa. Muille käyttöjärjestelmille julkaistut versiot eivät ole olleet yhtä suoritusvarmoja, ja ne ovat saattaneet lisätä tehonkulutusta. Näitä ongelmia ei ole voinut korjata muuten kuin odottamalla Adoben julkaisemaa korjausta. Tästä syystä Apple päätti syrjäyttää Flash-selainlaajennuksen tarjoamat tekniset mahdollisuudet suuntaamalla kehitystyön avoimiin HTML5-pohjaisiin ratkaisuihin [37]. Tällä tavoin sovelmat lopulta nopeuttivat avointen HTML-tekniikalla toimivien vaihtoehtojen kehitystä.

2.3.1 HTML5 Canvas

Applen tuki avoimille HTML5-pohjaisille ratkaisuille selainliitännäisten sijaan antoi suuren sysäyksen HTML5 Canvas -tekniikan kehitykselle ja leviämiselle. Canvas-elementti lähti kehittymään alun perin Applen Safari-selaimeen 2004 kehittämästä ominaisuudesta. Canvas-tekniikka liitettiin myöhemmin mukaan WHATWG-ryhmän kehittelemään HTML5-määrittelyyn ja edelleen lopulliseen W3C-ryhmän HTML5-standardiin.

HTML5-standardin määrittelemä Canvas-elementti on bittikarttagrafiikan esittämiseen tarkoitettu HTML-elementti. Canvas eroaa tavallisista HTML-elementeistä siten, että sen sisälle voidaan piirtää grafiikkaa vapaasti pikselitasolla. Tavallisten HTML-elementtien ulkoasua voidaan muokata CSS-tyylimäärittelyjen avulla, jotka rajoittuvat ennalta määriteltyihin arvoihin. Canvas-elementin sisälle piirrettäviin sisältöihin ei voida vaikuttaa CSS-määrittelyillä, vaan sisältöä muokataan erillisen JavaScript-rajapinnan kautta. Rajapinnan kautta suoritetaan yksinkertaisia piirtokomentoja, kuten esimerkissä 2.3. [1, 38]

```
1
2 var canvas = document.getElementById('testCanvas');
3 var context = canvas.getContext('2d');
4 var centerX = canvas.width / 2;
5 var centerY = canvas.height / 2;
6 var radius = 50;
7
8 context.beginPath();
```

[3]Youtube http://youtube.com

```
9  context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
10 context.fillStyle = 'red';
11 context.fill();
12 context.lineWidth = 5;
13 context.strokeStyle = '#000000';
14 context.stroke();
```

Esimerkki 2.3: Canvas-elementin API-komennot, jotka piirtävät punaisen ympyrän.

HTML5 Canvas -tekniikkaa käytetään nykyään monipuolisesti erilaisten sisältöjen visualisoinnissa. Peli- ja viihdesovellusten lisäksi Canvas-elementtiä on käytetty monenlaisissa hyötysovelluksissa, joissa ei ole ollut mahdollista käyttää selainlaajennuksia. Esimerkiksi monien karttasivustojen mobiiliversiot ovat hyötyneet Canvas-elementistä: sen tehokas piirtotekniikka on poistanut tarpeen piirtää kartta valmiiksi kuviksi palvelimella [39]. Grafiikan esittämisen lisäksi Canvas-elementin käyttöä on tutkittu esimerkiksi laiteympäristön tunnistamisessa, jossa testidatan piirtojäljen perusteella voidaan tunnistaa käytettävä laite, käyttöjärjestelmä ja selain [40].

HTML5 Canvas -elementille on erittäin laaja tuki nykysyisin käytössä olevissa internetselaimissa: Caniuse.com-sivuston[4] arvion mukaan helmikuussa 2016 Canvas-elementille oli täysi tuki 91,58 %:ssa selaimista ja osittainen tuki 4,81 %:ssa selaimista, taulukko A.3. Laaja tuki osoittaa, että Canvas-tekniikan asema on vakiintunut. Selaimet ja niiden prosentuaaliset markkinaosuudet on listattu taulukossa A.1.

2.3.2 WebGL

WebGL (Web Graphics Library) on 3D-grafiikan piirtämiseen tarkoitettu tekniikka, joka toimii Canvas-elementin sisällä. WebGL sisältää tavallisen Canvas-elementin tavoin joukon JavaScript-kutsuja, joiden avulla on mahdollista ohjata ruudulle piirrettävää grafiikkaa. WebGL ei ole osa virallista HTML5-stardardia, vaan sitä kehittää voittoa tavoittelematon Khronos Group[5], jonka jäseniä ovat selainvalmistajat Apple, Google ja Mozilla. [41, 42]

Suurin osa nykyisistä selaimista tukee WebGL-tekniikkaa ainakin osittain: Caniuse.com-sivuston mukaan helmikuussa 2016 täysi WebGL-tuki oli 57,52 %:ssa selaimista ja osittainen tuki 27,9 %:ssa selaimista, taulukko A.4.

Osa tässä tutkimuksessa läpikäytävistä Canvas-kirjastoista tukee WebGL-tekniikan käyttämistä vaihtoehtoisena 2D-grafiikan piirtotekniikkana tavallisen Canvas-elementin sijaan. WebGL-tekniikan käyttäminen on

[4]Caniuse.com http://caniuse.com

[5]Khronos Group https://www.khronos.org

suoritustehokkuuden kannalta hyödyllistä, koska se pystyy käyttämään grafiikkaprosessorin tarjoamaa laitekiihdytystä grafiikan piirtämisessä. WebGL-tekniikka ei kuitenkaan ole käytettävissä kaikissa laitteissa, koska selaimen lisäksi laitteen on tuettava sitä.

Tässä tutkimuksessa on keskitytty tavallista Canvas-elementtiä käyttäviin kirjastoihin, koska WebGL-tekniikan tuki ei ole vielä yhtä laaja kuin tavallisen Canvas-elementin. Tutkittavissa kirjastoissa mahdollisuus käyttää WebGL-tekniikkaa lasketaan eduksi tulevaisuuden käytettävyyden kannalta.

HTML5 Canvas -tekniikan käyttövalmius selaimissa	
täysin tuettu	91,58 %
osittain tuettu	4,81 %
yhteensä	96,39 %

Taulukko 2.1: Caniuse.com-sivuston julkaisema HTML5 Canvas -tekniikan tuki helmikuussa 2016. Vertailu sisältää selaimet, jotka muodostavat 97,41 % selainten markkinaosuuksista helmikuussa 2016. Tarkemmat tiedot on esitetty kuvassa A.3

WebGL -tekniikan käyttövalmius selaimissa	
täysin tuettu	57,52 %
osittain tuettu	27,90 %
yhteensä	85,42 %

Taulukko 2.2: Caniuse.com-sivuston julkaisema WebGL-tekniikan tuki helmikuussa 2016. Vertailu sisältää selaimet, jotka muodostavat 97,41 % selainten markkinaosuuksista helmikuussa 2016. Tarkemmat tiedot on esitetty kuvassa A.4

Luku 3

Ohjelmointikirjastot ja ohjelmakoodin uudelleenkäyttö

Vuonna 1968 Nato Software Engineering Conference -tapahtumassa julkaistussa artikkelissa Mass Produced Software Components käsiteltiin ensimmäisen kerran ohjelmakoodin uudelleenkäyttöä ohjelmistokomponenttien avulla. Ajatuksena oli luoda koodista modulaarista siten, että yleiskäyttöiset osat ohjelmakoodista olisivat ostettavissa valmiina aliohjelmina.[43]

Vuoden 1968 jälkeen ohjelmakoodin uudelleenkäyttöä ohjelmistokehityksessä on tutkittu ja myös käytetty paljon. Tavoitteena on ollut vähentää ohjelmistokehityksen kuluja ja parantaa ohjelmistojen laatua. Uudelleenkäytössä on sekä hyötyjä ja haittoja, jotka tulee tiedostaa ennen ohjelmistoprojektin aloitusta. Koodin uudelleenkäyttötavan valinta on usein kompromissi, johon vaikuttavat käytössä olevat resurssit, aikataulu, ohjelmiston ylläpitokustannukset ja ohjelmiston jatkokehityssuunnitelmat. [44, 45]

Ohjelmakoodin uudelleenkäytöllä saavutetaan usein seuraavia hyötyjä ohjelmistoprojekteissa:

- **Pienemmät kustannukset.** Ohjelmistoprojektissa tarvitsee kirjoittaa kokonaismäärältään vähemmän ohjelmakoodia, jolloin ohjelmiston kehityskulut usein pienenevät. Myös testaamiseen ja ylläpitoon tarvittavat resurssit pienenevät.

- **Pienempi riski.** Osa ohjelmistoprojektin ohjelmakoodista on jo olemassa ja sen kustannukset tiedetään ennalta. Tämä helpottaa ohjelmiston kustannusten arviointia ja pienentää arvion virhemarginaalia.

- **Nopeutunut kehitysaikataulu.** Osa ohjelmistosta on jo valmiina, jolloin kehitysaikataulu on usein nopeampi.

- **Parantunut laatu.** Uudelleenkäytettävä osa ohjelmistosta on jo kehitetty ja sitä on testattu kehityksen yhteydessä. Mahdollisia virheitä on usein vähemmän kuin uudessa ohjelmakoodissa.
- **Standardien käyttäminen.** Uudelleenkäytettävät osat on yleensä ohjelmoitu standardien ja yleisten käytäntöjen mukaisesti. Tällöin uudelleenkäytettäviä osia hyödyntävät ohjelmoijat tuntevat usein tekniikat ja osaavat käyttää niitä helpommin.
- **Erityisosaaminen.** Uudelleenkäytettävän osan ovat ohjelmoineet usein henkilöt, jotka ovat erikoistuneet kyseiseen tekniikkaan. [45]

Nykyaikaisessa ohjelmistokehityksessä ohjelmakoodin uudelleenkäyttö tehdään usein oliopohjaisten ohjelmointikirjastojen tai ohjelmistokehysten avulla. Verkkosivujen ja selainpohjaisten ohjelmistojen kehityksessä ne ovat nousseet lähes samaan asemaan kuin erilliset ohjelmointikielet. Paljon toiminnallisuuksia sisältävien ohjelmistoprojektien ohjelmoinnissa voidaan säästää paljon aikaa ja resursseja käyttämällä kehityksessä kirjastoja tai kehyksiä [45]. Esimerkiksi käyttöliittymäohjelmoinnissa käytettävällä Angular.js-ohjelmistokehyksellä pystyttiin Google Feedback -projektissa kirjoittamaan uudestaan kolmessa viikossa käyttöliittymä, jonka tekeminen oli alun perin kestänyt 6 kuukautta. Samalla koodirivien määrä supistui 17 000 rivistä 1 500 riviin. [46]

Ohjelmointikirjastojen ja ohjelmistokehysten tehokas käyttö vaatii sen, että kehittäjät tuntevat hyvin niiden toiminnan. Useimmat niistä vaativat käyttäjältä toiminnallisuuden opettelua ennen käyttöönottoa [47]. Tällöin uusien kirjastojen opettelu vie aikaa kehittäjiltä ja nostaa kehityskuluja. Ohjelmistoprojekteissa on hyödyllistä, että kehittäjillä on käytössä mahdollisimman hyvin yleiskäyttöiseen kehitykseen soveltuvia ohjelmistokirjastoja. Tällöin kehittäjän tarvitsee hallita pienempi määrä erilaisia tekniikoita ja uusien henkilöiden liittäminen projektiin vaatii vähemmän perehdytysaikaa.

Termejä ohjelmointikirjasto ja ohjelmistokehys käytetään jonkin verran ristiin HTML5 Canvas -ohjelmointiin tarkoitettujen ohjelmistojen yhteydessä. Termiä framework, joka viittaa ohjelmistokehykseen käytetään, myös sellaisista ohjelmistoista, jotka ovat enemmänkin ohjelmistokirjastoja.

Ohjelmointikirjasto voidaan määritellä siten, että sitä käytettäessä varsinainen ohjelmakoodi kutsuu kirjaston luokkia ja funktioita. Ohjelmistokehyksessä ohjelmakoodi kirjoitetaan sisälle kehyksen määrittelemään rakenteeseen.

Tässä tutkimuksessa keskitytään tarkastelemaan HTML5 Canvas -elementin ohjelmointiin käytettäviä ohjelmointikirjastoja ja ohjelmistokehyksiä, joita voidaan käyttää myös kirjaston tavoin. Tämä rajaus on tehty

siitä syystä, että laajemmissa ohjelmistoprojekteissa saattaa olla käytössä muita ohjelmistokehyksiä, jotka vaikuttavat ohjelmiston rakenteeseen. Tällöin Canvas-elementin ohjelmistokehyksen vaatimat ohjelmiston rakenteet voivat aiheuttaa ristiriitoja. Ohjelmointikirjastomallinen lähestymistapa Canvas-elementtien ohjelmoinnissa luo vähemmän rajoituksia ja riippuvuuksia kokonaisuuden kannalta.

Luku 4

Testi- ja tutkimusmenetelmät

Tutkimuksessa tarkasteltavista kirjastoista valitaan tutkimustulosten perusteella yleiskäyttöiseen HTML5 Canvas -elementin ohjelmointiin parhaiten soveltuvat kirjastot. Tutkimuksessa pyritään tunnistamaan ne kirjastot, jotka täyttävät seuraavat vaatimukset:

- Kirjasto täyttää luvussa 4.1 Esikarsinta listatut ominaisuudet.
- Kirjasto täyttää tekniset vaatimukset, jotka on esitelty luvussa 4.2 Tekniset ominaisuudet.
- Kirjaston kehitys ja ylläpito on aktiivista ja jatkuvaa. Tätä vaatimusta esitellään tarkemmin luvussa 4.3 Kehitys- ja käyttäjäyhteisön aktiivisuus.
- Kirjaston graafinen suorituskyky on kilpailukykyinen verrattuna muihin tutkimuksessa tarkasteltaviin kirjastoihin. Suorituskyvyn mittareita selostetaan tarkemmin luvussa 4.4 Tehokkuus.

4.1 Esikarsinta

Tutkimuksen ensimmäisessä vaiheessa listataan kaikki tutkimukseen valitut kirjastot ja tehdään niille esikarsinta. Esikarsinnassa arvioidaan joitain kirjastojen teknisiä ominaisuuksia ja niiden kehitysprojektien ominaisuuksia. Tarkasteltavat ominaisuudet on valittu siten, että niiden puuttuminen tai täyttymättä jääminen johtavat suoraan kirjaston hylkäämiseen tarkemmasta tutkimuksesta.

Esikarsinnassa tarkastellaan kirjastoja seuraavista näkökulmista:

- Onko kirjaston kehitysprojektiin tullut päivityksiä viimeisen kuuden kuukauden aikana?

Jos kirjaston kehitysprojektiin ei ole tullut päivityksiä viimeisen kuuden kuukauden aikana, sen kehitysyhteisö ei vaikuta aktiiviselta. Aktiivisissa kehitysprojekteissa käyttäjät ehdottavat uusia ominaisuuksia, raportoivat virheitä ja lähettävät kehittämiään uusia ominaisuuksia arvioitaviksi projektista vastaaville henkilöille. Jos kirjastoa ei ylläpidetä, sen käyttämiseen sitoutuminen on riskialtista pitkällä aikavälillä. Kirjasto hylätään, jos siihen ei ole tullut päivityksiä viimeisen kuuden kuukauden aikana.

- Onko ohjelmointikirjaston kehitysprojekti avoimessa versionhallintajärjestelmässä?

Tutkitaan, onko ohjelmistokirjaston koodi tallennettu avoimeen versionhallintajärjestelmään. Ohjelmointikirjastolta vaaditaan, että se on tallennettu johonkin avoimeen versionhallintajärjestelmään, jotta ohjelmointikirjastoa käyttävä ohjelmoija voi tutustua paremmin kirjaston kehityshistoriaan sekä ehdottaa mahdollisia muutoksia tai korjauksia. Esimerkiksi GitHub- tai BitBucket-versionhallinta tai vastaavat Git-versionhallintajärjestelmät lasketaan soveltuviksi ympäristöiksi. Jos kirjastolla ei ole julkista versionhallintaa, se hylätään.

- Salliiko kirjaston avoimen lähdekoodin lisenssi kaupallisen käytön?

Suurin osa avoimen lähdekoodin HTML5 Canvas -elementin ohjelmistokirjastoista on julkaistu vapaaseen käyttöön MIT- tai GPL-lisenssillä. Ohjelmointikirjastojen lisenssiehdot tarkistetaan, jotta kirjastoa käyttävällä ohjelmistokehittäjällä on tarvittavat oikeudet ohjelmistojen kehittämiseen. Oikeuksien tulee sallia kaikenlainen kaupallinen käyttö. Tässä tutkimuksessa keskitytään vapaisiin avoimen lähdekoodin ohjelmointikirjastoihin. Ohjelmointikirjasto hylätään, jos sitä ei ole julkaistu avoimella lisenssillä ja valittu lisenssi ei mahdollista kirjaston kaupallista käyttöä.

- Onko kirjastolla olemassa dokumentaatio sen rakenteesta ja ohjeet sen käyttämiseen?

Kirjaston rakenteen tekniset kuvaukset ja käytännön esimerkit ovat tärkeitä kirjaston toiminnan opettelussa. Dokumentaation puuttuminen vaikeuttaa kirjaston käyttöönottoa ja lisää siihen kuluvaa aikaa ohjelmistoprojekteissa. Lisäksi dokumentaation puuttuminen tai sen huono laatu ovat merkkejä kirjaston kehitysprojektin puutteista. Kirjasto hylätään, jos sen dokumentaatio arvioidaan puutteelliseksi.

- Sisältääkö kirjasto teknisiä riippuvuuksia?

Tarkoituksena on löytää kirjasto, jota voidaan käyttää muiden teknisten kirjastojen ja ohjelmistoratkaisuiden ohessa HTML5 Canvas -elementin ohjelmointiin. Kirjasto tulee pystyä liittämään osaksi suurempaa kokonaisuutta ohjelmistoprojekteissa. Kirjaston tulee vaatia vain JavaScript-kielen käyttämistä ilman riippuvuuksia muista kirjastoista. Kirjasto hylätään, jos sen kehitysympäristö tai kirjaston toiminta sisältävät teknisiä riippuvuuksia.

- Onko kirjasto soveltuva yleiseen HTML5 Canvas -elementin graafiseen ohjelmointiin ohjelmistoprojekteissa?

Tarkastellaan kirjaston käyttötarkoitusta ja sen rakennetta. Jos kirjasto on tarkoitettu vain yhteen tarkoitukseen, se hylätään.

4.2 Tekniset ominaisuudet

Tutkimuksessa tarkastellaan esikarsinnan läpäisseiden HTML5 Canvas -ohjelmointikirjastojen teknisiä ominaisuuksia ja vertaillaan kirjastoja niiden perusteella. Teknisten ominaisuuksien tarkastelulla varmistetaan, että tutkimuksessa mukana olevat kirjastot täyttävät niille asetetut perusedellytykset grafiikan ohjelmoinnissa. Osa ominaisuuksista on asetettu pakollisiksi, jolloin kirjaston on täytettävä nämä ominaisuudet. Loput ominaisuudet ovat valinnaisia, eikä niiden puuttuminen johda kirjaston hylkäämiseen.

4.2.1 Pakolliset ominaisuudet

Kirjastojen on täytettävä seuraavat tekniset vaatimukset:

- Tukeeko kirjasto vektorigrafiikan piirtämistä?

Kirjastolla tulee olla mahdollista piirtää vektorigrafiikalla samoja peruselementtejä kuin Canvas-elementillä ilman erillistä ohjelmointikirjastoa.

- Tukeeko kirjasto bittikarttagrafiikan piirtämistä?

Kirjastolla tulee olla mahdollista piirtää bittikarttagrafiikkaa, jonka lähdemateriaali on ladattavissa erillisistä kuvatiedostoista.

- Tukeeko kirjasto graafisten objektien muodostamista?

Kirjastolla tulee olla mahdollista muodostaa graafisia objekteja, joissa voidaan yhdistää vektorigrafiikkaa, bittikarttagrafiikkaa sekä tekstiä. Yksittäistä graafisia objektia tulee olla mahdollista käsitellä yhtenä kokonaisuutena, jolloin ohjelmoijan ei tarvitse itse pitää huolta yksittäisten, matalan tason graafisten elementtien päivittämisestä jokaisella ruudunpäivitystapahtumalla.

- Tukeeko kirjasto Sprite sheet -animaatioiden toteuttamista?

Kirjastolla tulee olla mahdollista muodostaa Sprite sheet -animaatioita ilman erillistä ohjelmointia. Sprite sheet -tekniikassa ladataan yksi kuvatiedosto, joka sisältää bittikartta-animaation tarvitsemat ruudut. Kirjaston on pystyttävä muodostamaan Sprite sheet -animaatio kirjaston peruskomennoilla.

- Tukeeko kirjasto Tween-animointia?

Kirjastolla tulee olla mahdollista animoida graafisia objektien ominaisuuksia Tween-tekniikalla. Tämä tarkoittaa, että animoitavalle ominaisuudelle asetetaan kohdearvo ja animaation kesto. Ominaisuus voi olla esimerkiksi graafisen objektin paikka, väri tai alpha-arvo.

- Tukeeko kirjasto graafisten objektien törmäyksen tarkistusta?

Kirjaton tulee tukea vähintään perusmuotoista graafisten objektien törmäyksen tarkistusta. Tämä tarkoittaa, että kirjaston tulee pystyä tarkistamaan kahden graafisen objektin törmäys vähintään MBR-tekniikan (Minimum Bounding Rectangle) avulla. MBR-tekniikassa elementin alueeksi lasketaan sen ympärille piirretty pienin mahdollinen suorakulmio. Kahden elementin törmäys lasketaan tarkastelemalla näiden suorakulmioiden törmäystä. Tekniikka on laskennallisesti yksinkertaisempi, kuin jos tarkasteltaisiin graafisen objektin jokaista yksittäistä elementtiä erikseen. Monimutkaisempi törmäyksen tarkistus elementin oikeiden mittojen avulla lasketaan kirjaston eduksi, mutta sitä ei edellytetä.

- Tukeeko kirjasto tekstielementtejä?

Kirjastolla tulee olla mahdollista luoda tekstielementtejä. Näiden tekstielementtien tulee toimia samalla tavalla kuin muidenkin graafisten elementtien. Lisäksi tekstin fontti ja muut ominaisuudet, kuten väri ja koko, pitää pystyä määrittelemään.

- Tukeeko kirjasto hiirellä tehtäviä toimintoja?

Kirjaston tulee tukea hiirellä annettavia komentoja. Perustoimintojen, kuten hiiren kursorin tunnistaminen elementin päällä sekä graafisten elementtien raahaaminen, täytyy toimia.

- Tukeeko kirjasto kosketusnäytöllä tehtäviä komentoja?

Kirjaston tulee toimia kosketusnäytöllisten laitteiden kanssa. Sen pitää tunnistaa perustoiminnot kuten napautus ja raahaus. Multitouch-ominaisuudet eivät pakollisia, mutta lasketaan kirjaston eduksi.

4.2.2 Valinnaiset ominaisuudet

Seuraavat tekniset ominaisuudet lasketaan eduksi kirjastolle, mutta niiden puuttuminen ei johda kirjaston hylkäämiseen:

- Sisältääkö kirjasto tunnettuja optimointimenetelmiä?

Listataan kirjaston sisältämät graafisen piirtonopeuden optimoinnit, joita kirjastossa voidaan käyttää niiden soveltuessa ohjelman rakenteeseen. Näitä optimointeja ovat:

Cache as Bitmap -optimointi. Monimutkaisesta graafisesta elementistä otetaan bittikarttakopio ja käytetään sitä nopeuttamaan ruudunpäivitystä. Tämä vähentää erillisiä matalan tason piirtokomentoja varsinkin, jos elementti on koostuu useista erillisistä elementeistä. Optimointi hyödyttää vain, jos piirrettävä elementti ei muutu ruudunpäivitysten aikana.

Usean Canvas-elementin käyttö päällekkäisinä piirtotasoina. Piirretään graafiset elementit erillisille päällekkäisille Canvas-elementeille. Ruudunpäivityksessä tarvitsee tällöin piirtää uudestaan vain ne Canvas-elementit, joiden sisältämät graafiset elementit ovat muuttuneet. Tämä optimointi vähentää jokaisen ruudunpäivityksen aikana suoritettavia piirtokomentoja. Optimoinnista on hyötyä, jos osa elementeistä pysyy muuttumattomana.

Ruudun päivityksen estäminen, jos luodaan uusi elementti tai poistetaan vanha. Optimoinnilla voidaan vähentää kertoja, jolloin ruutu piirretään uudestaan lisättäessä ja poistettaessa elementtejä.

- Tukeeko kirjasto vaihtoehtoista WebGL-piirtotekniikkaa?

Selvitetään, tukeeko ohjelmointikirjasto vaihtoehtoista WebGL-tekniikkaa grafiikan piirtämisessä. WebGL-tuki parantaa tehokkaamman piirtonopeuden ansiosta käyttökokemusta niillä käyttäjillä, joiden päätelaite ja selain tukevat sitä.

4.3 Kehittäjä- ja käyttäjäyhteisön aktiivisuus

Tarkoituksena on tutkia, miten aktiivisia ohjelmistokirjaston kehittäjät ja käyttäjät ovat. Jos kehittäjäyhteisö ei vaikuta aktiiviselta, mahdollisten virheiden korjaaminen ja kirjaston jatkokehitys ei todennäköisesti ole nopeata.

Jos kirjastossa ei tapahdu kehitystä, sen käyttäminen esimerkiksi kaupallisissa tuotteissa ei ole toivottavaa. Käyttäjäyhteisön aktiivisuus kertoo siitä, miten elinvoimainen ohjelmointikirjaston kehitys tulee jatkossa olemaan. Jos kirjastolla ei ole aktiivista käyttäjäjoukkoa, sen kehitys on todennäköisesti jatkossa epävarmaa.

Kehittäjäyhteisön aktiivisuutta arvioidaan versionhallintajärjestelmästä saatavilla numeerisilla arvoilla. Luvut ovat suuntaa antavia, koska eri kirjastojen kehitysprojektit ovat olleet olemassa eri pituisen ajan ja niiden kehitys on jokaisessa projektissa edennyt eri tavalla. Luvuista voidaan kuitenkin saada viitteitä siitä, mihin kirjaston kehitys on menossa: esimerkiksi jos kirjaston koodipohjaan on tehnyt päivityksiä viimeisen vuoden aikana 10 henkeä, voidaan olettaa, että kirjaston kehitysprojekti on jatkossa aktiivisempi kuin yhden ihmisen ylläpitämänä. Versionhallinnasta on kerätty lukuja sekä koko projektin että viimeisen vuoden ajalta. Viimeisen vuoden sisältä tarkastellaan versionhallintaan tallennettujen päivitysten määrää sekä päivityksiä tehneiden henkilöiden määrää.

Kehittäjäyhteisön aktiivisuutta tutkitaan analysoimalla Git-versiohallinnan kautta saatuja seuraavia lukuja:

- Milloin projektin versionhallinta on aloitettu?

Tarkastellaan, miten pitkään kirjastoa on kehitetty versionhallinnassa. Koko projektin ajalta saadut luvut tulee suhteuttaa projektin kestoon: osa kirjastoista on ollut kehitteillä useita vuosia, kun taas osalla on takana alle vuosi kehityshistoriaa.

- Kuinka monta kertaa versionhallintaan on lisätty koodia?

Tarkastellaan kuinka monta kertaa projektissa on lisätty koodia versionhallintaan (Commits) viimeisen vuoden sekä koko projektin aikana.

- Kuinka monen eri henkilön koodia on lisätty projektin versionhallintaan?

Tarkastellaan, kuinka moni henkilö on ollut mukana tuottamassa koodia projektiin (Contributors). Kehittäjäjoukon laajuutta tarkastellaan sekä koko projektin että viimeisen vuoden ajalta. Luvuista selviää karkeasti, kuinka laaja kehittäjäjoukko on kiinnostunut projektin eteenpäin viemisestä.

- Kuinka monta kertaa projektin versionhallintaan lähetettyjä koodimuutoksia on hyväksytty tai hylätty?

Tarkastellaan, kuinka monta kertaa projektiin ehdotettuja koodimuutoksia merkitty käsitellyiksi (Pull requests closed). Tämä luku kertoo, kuinka monta muutosehdotusta projektin vetäjät ovat joko hyväksyneet tai hylänneet. Luku antaa viitteitä myös siitä, miten aktiivisesti kehittäjäyhteisö on lähettänyt muutoksia.

- Kuinka monta versionhallintaan lähetettyä ehdotettua koodimuutosta on käsittelemättä?

Tarkastellaan, kuinka monta pääkehittäjien ulkopuolelta tullutta koodimuutosta projektissa on käsittelemättä (Pull requests open). Yhdessä käsiteltyjen ehdotusten kanssa (Pull request closed) luvuista voi päätellä suuntaa antavasti, miten tehokkaasti projektin versionhallintaa hallinnoivat pääkehittäjät käsittelevät koodiehdotukset. Jos käsittelemättömiä, avoimia koodimuutoksia on paljon, pääkehittäjät eivät suhtaudu niihin kovin aktiivisesti.

- Kunka monta projektille esitettyä kysymystä on merkitty käsitellyksi?

Tarkastellaan, kuinka monta pääkehittäjille esitettyä kysymystä on merkitty käsitellyksi (Issues closed). Luku kertoo suuntaa antavasti siitä, miten aktiivisesti pääkehittäjät ratkaisevat heille esitettyjä kysymyksiä.

- Kunka monta avointa kysymystä projektilla on odottamassa käsittelyä?

Tarkastellaan, kuinka monta pääkehittäjille suunnattua kysymystä ei ole vielä merkitty käsitellyksi (Issues open). Yhdessä käsitellyiksi merkittyjen kysymysten (Issues closed) kanssa luku kertoo suuntaa antavasti siitä, kuinka paljon kysymyksiä ja kommentteja projektissa on esitetty lisäominaisuuksista tai ohjelmistovirheistä.

- Kuinka moni on merkinnyt projektin versionhallinnan seurattavaksi itselleen? (Watchers)

Luku kertoo siitä, kuinka moni on kiinnostunut seuraamaan muutoksia projektin versionhallinnassa.

- Kuinka monta kertaa projektin koodeista on lähdetty kehittämään omaa versiota? (Forks)

Luku kertoo siitä, miten moni on kiinnostunut kehittämään kirjastosta oman versionsa. On kuitenkin tärkeää huomioida, että osa kirjastoista suosittelee käyttäjille omien versioiden luomista ja osa taas kehottaa käyttämään kirjastoa sellaisenaan.

4.4 Tehokkuus

Referenssiohjelman tehokkuutta tutkitaan mittaamalla keskimääräinen ruudunpäivitysnopeus eri testitapauksissa. Tehokkuutta testataan samalla testikoneella ja samalla selaimella. Testiohjelmille ohjelmoidaan yleiskäyttöinen ruudunpäivityksen mittaava ohjelmakoodi, jonka sisälle liitetään aina eri kirjastolla toteutettu testiohjelma.

Testeissä ei aktivoitu kirjastoissa mahdollisesti käytettävissä olevia optimointeja. Tarkoituksena oli tutkia kirjastojen piirtonopeutta yleisellä tasolla ja välttää yksittäiseen graafiseen ongelmaan sopivien optimointien vaikutus.

Kirjastoille ajetaan seuraavat testit:

- vektorigrafiikalla toteutettujen liikkuvien sprite-elementtien piirtotehokkuus

Testataan kirjaston piirtonopeutta erilaisilla määrillä sprite-elementtejä. Tarkoituksena on selvittää, onko kirjastoissa eroja korkeamman tason piirtoelementtien hallinnassa ja onko elementtien piirtoa tältä osin optimoitu. Ilman optimointeja kirjasto joutuu piirtämään jokaisessa ruudunpäivityksessä suuren joukon vektorigrafiikkaelementtejä.

- bittikarttagrafiikalla toteutettujen liikkuvien ja animoitujen elementtien piirtotehokkuus

Testataan kirjaston piirtonopeutta animoidulla, bittikarttagrafiikalla toteutetuilla sprite-elementeillä. Kirjasto joutuu piirtämään jokaisessa piirtoalueen päivityksessä uudestaan sprite-elementtien sisältämän bittikarttagrafiikan.

4.4.1 Muistinkäsittely

Referenssiohjelman muistinkäyttöä tutkitaan mittaamalla Javascript-suorituksen tarvitseman muistin määrää selaimen seurantatyökalujen avulla. Tarkoituksena on tutkia, tapahtuuko suorituksen aikana muistivuotoja ja miten paljon muistia ohjelmointikirjastot kuluttavat ajonaikaisesti.

Luku 5

Tutkimuskohteiden esikarsinta

Tutkimukseen valituille ohjelmointikirjastoille tehdään tässä luvussa esikarsinta. Tarkoituksena on tutkia, mitkä kirjastot täyttävät esikarsinnassa asetetut vähimmäisvaatimukset. Esikarsinnan läpäisseet kirjastot valitaan mukaan luvussa 6 tehtävään ominaisuuksien tarkempaan läpikäyntiin.

Tutkimuksessa tarkastellaan tarjolla olevat avoimen lähdekoodin HTML5 Canvas -ohjelmistokirjastot. Tutkimukseen mukaan otettujen kirjastojen lista on muodostettu käymällä läpi versionhallintasivustoja, kuten GitHubia[1] ja BitBucketia[2], ja ohjelmointiaiheisia keskustelufoorumeita, kuten Stackoverflow'ta[3]. Lisäksi on tutustuttu HTML5 Canvas -elementtiä käyttävien verkkosivustojen käyttämiin ohjelmointikirjastoihin. Tutkimuksessa käsiteltäviksi kirjastoiksi valittiin taulukossa 5.1 listatut kirjastot.

kirjasto	**lisenssi**	**kotisivu**
Artisan.js	MIT, GPLv3	`http://artisanjs.com/`
bHive	GPLv2	`http://bhivecanvas.com/`
CAAT (Canvas Advanced Animation Toolkit)	MIT	`https://hyperandroid.github.io/CAAT/`
CAKE (Canvas Animation Kit Experiment)	MIT	`http://glimr.rubyforge.org/cake/`
Cango	Public Domain	`http://www.arc.id.au/`

[1]Github.com `https://github.com`
[2]Bitbucket `https://bitbucket.org`
[3]Stackoverflow `https://stackoverflow.com`

Canto.js	MIT	https://code.google.com/archive/p/canto-js/
Canvas Engine	MIT	http://canvasengine.net/
Canvas Query	MIT	http://canvasquery.com/
Cocos2D-x HTML5	MIT	http://www.cocos2d-x.org/
CraftyJS	MIT GPL	http://craftyjs.com/
Doodle.js	BSD	https://github.com/lamberta/doodle-js/
Easel.js	MIT	http://www.createjs.com/easeljs/
Enchant.js	MIT	http://enchantjs.com/
Fabric.js	MIT	http://fabricjs.com/
Goo.js	MIT	http://www.storminthecastle.com/projects/goo.js/
Graphics2D.js	MIT LGPL	http://graphics2d.js.org/
Gury.js	MIT	https://github.com/rsandor/gury/
HTML Canvas lib	MIT	http://html-canvas-lib.sourceforge.net/
HTML5 Canvas Library	MIT	https://canvastoolkit.codeplex.com/
Isogenic Game Engine	MIT	http://www.isogenicengine.com/
jCanvaScript	GPLv2	http://jcscript.com/
Kinetic.js	MIT GPLv2	http://kineticjs.com/
Kiwi.js	MIT	http://www.kiwijs.org/
Konva.js	MIT GPLv2	https://konvajs.github.io/
Layered Canvas Library	GPLv3	https://code.google.com/archive/p/layered-canvas-library/

LibCanvas	MIT LGPL	http://libcanvas.com/
Lime.js	Apache License 2.0	http://www.limejs.com/
Melon.js	MIT	http://melonjs.org/
Mootools Canvas Lib	MIT	http://mootools.net/forge/p/mootools_canvas_lib
oCanvas.js	MIT	http://ocanvas.org/
Panda.js	MIT	http://www.pandajs.net/
Paper.js	MIT	http://paperjs.org/
Phaser.io	MIT	http://phaser.io/
Pixi.js	MIT	http://www.pixijs.com/
Playground.js	MIT	http://playgroundjs.com/
Processing.js	MIT	http://processingjs.org/
Quintus	MIT GPLv2	http://www.html5quintus.com/
SceneGraph.js	MIT	https://gwennaelbuchet.github.io/SceneGraph.js/
Scrawl.js	MIT	http://scrawl.rikweb.org.uk/
Stage.js	MIT	http://piqnt.com/stage.js/
Two.js	MIT	https://jonobr1.github.io/two.js/
xc.js	BSD	https://github.com/fairfieldt/xcjs

Taulukko 5.1: Tutkimuksessa käsiteltävät HTML5 Canvas -ohjelmointikirjastot.

5.1 Päivitysajankohta

Kirjastojen versiohistoriaa tutkimalla hylätään taulukossa 5.2 listatut kirjastot. Näihin kirjastoihin ei ole tarkasteluajankohtana 24.5.2016 julkaistu päivityksiä edellisen kuuden kuukauden aikana.

kirjasto	**viimeisin päivitys**
Artisan.js	21.8.2011
bHive	11.8.2011
CAAT	2.7.2013
CAKE	2.7.2012
Canto.js	19.10.2010
Canvas Engine	11.4.2014
Doodle.js	23.11.2011
Goo.js	13.10.2013
Gury.js	9.8.2011
HTML Canvas lib	11.4.2011
HTML5 Canvas Library	10.1.2012
Isogenic Game Engine	18.9.2015
jCanvaScript	20.9.2012
Kinetic.js	15.1.2015
Kiwi.js	16.11.2015
Layered Canvas Library	14.10.2010
Lime.js	1.6.2015
Mootools Canvas Lib	31.5.2010
Panda.js	17.2.2015
SceneGraph.js	26.5.2015
xc.js	15.7.2015

Taulukko 5.2: Päivitysajankohdan perusteella tutkimuksessa hylätyt kirjastot. Tarkasteluajankohta 24.5.2016.

5.2 Versionhallinta

Versionhallintatietoja tarkastellaan niiden kirjastojen osalta, joita ei hylätty luvussa 5.1. Puutteellisen versionhallinnan perusteella hylättiin yksi kirjasto (taulukko 5.3).

kirjasto	**versionhallinta**
Cango	ei julkista versionhallintaa

Taulukko 5.3: Julkisen versionhallinnan puuttumisen perusteella tutkimuksessa hylätyt kirjastot. Tarkasteluajankohta 24.5.2016.

5.3 Dokumentaatio

Kirjastojen dokumentaatiota tarkastellaan niiden kirjastojen osalta, joita ei hylätty luvuissa 5.1 tai 5.2. Riittämättömän dokumentaation tai sen puuttumisen takia hylätyt kirjastot on listattu taulukossa 5.4.

kirjasto	**dokumentaatio**
Canvas Query	ei riittävän tarkkaa dokumentaatiota
Enchant.js	osa dokumentaatiosta on julkaistu vain japaniksi
Graphics2D.js	dokumentaatio puuttuu
LibCanvas	dokumentaatio puuttuu
Playground.js	ei riittävän tarkka dokumentaatio
Stage.js	ei riittävän tarkka dokumentaatio

Taulukko 5.4: Dokumentaation puuttumisen tai huonon laadun perusteella tutkimuksessa hylätyt kirjastot. Tarkasteluajankohta 24.5.2016.

5.4 Tekniset riippuvuudet

Kirjastojen teknisiä riippuvuuksia tarkastellaan niiden kirjastojen osalta, joita ei hylätty luvuissa 5.1, 5.2 tai 5.3. Teknisten riippuvuuksien takia hylättiin yksi kirjasto (taulukko 5.5).

kirjasto	**Tekninen riippuvuus**
Phaser.io	käyttää pixi.js kirjastoa grafiikan piirtämiseen

Taulukko 5.5: Teknisten riippuvuuksien takia hylätyt kirjastot.

5.5 Käyttötarkoitus ja rakenne

Kirjastojen käyttötarkoitusta ja rakennetta tarkastellaan niiden kirjastojen osalta, joita ei hylätty luvuissa 5.1, 5.2, 5.3 tai 5.4. Käyttötarkoituksen tai rakenteen takia hylätyt kirjastot on listattu taulukossa 5.6.

kirjasto	**hylkäysperuste**
Cocos2D-x HTML5	Cocos2D-x -kirjasto on tarkoitettu pelikehitykseen erilaisissa laiteympäristöissä. Kirjasto ei ole pääosin keskittynyt HTML5-tekniikkaan.
Processing.js	Ei sisällä graafisten elementtien käsittely objekteina.
Two.js	Tarkoitettu vain vektorigrafiikan esittämiseen. Ei sisällä bittikarttagrafiikan käsittelyä.

Taulukko 5.6: Käyttötarkoituksen tai rakenteen takia hylätyt kirjastot.

5.6 Esikarsinnan läpäisseet kirjastot

Taulukossa 5.7 on listattu ne ohjelmointikirjastot, jotka läpäisivät esikarsinnan, ja jotka otetaan seuraavaksi lähempään tarkasteluun.

kirjasto	**kotisivu**
CraftyJS	`http://craftyjs.com/`

Easel.js	http://www.createjs.com/easeljs/
Fabric.js	http://fabricjs.com/
Konva.js	https://konvajs.github.io/
Melon.js	http://melonjs.org/
oCanvas.js	http://ocanvas.org/
Paper.js	http://paperjs.org/
Pixi.js	http://www.pixijs.com/
Quintus	http://www.html5quintus.com/
Scrawl.js	http://scrawl.rikweb.org.uk/

Taulukko 5.7: Esikarsinnan läpäisseet HTML5 Canvas -ohjelmointikirjastot.

Luku 6

Tutkimuskohteiden ominaisuuksien tarkastelu

Tässä luvussa tarkastellaan esikarsinnan läpäisseiden kirjastojen teknisiä ominaisuuksia ja niiden projektinhallinnasta kerättyjä tietoja. Tarkastelun avulla rajataan pois kirjastot, jotka eivät täytä luvussa 4 määriteltyjä vaatimuksia.

6.1 Kirjastojen tekniset ominaisuudet

kirjasto	**vektoripiirto**	**bitmap-piirto**	**graafiset objektit**	**Sprite sheet -animaatiot**
CraftyJS	✓	✓	✓	✓
Easel.js	✓	✓	✓	✓
Fabric.js	✓	✓	✓	lisäosa[1]
Konva.js	✓	✓	✓	✓
Melon.js	✓	✓	✓	✓
oCanvas.js	✓	✓	✓	✓
Paper.js	✓	✓	✓	manuaalisesti[2]
Pixi.js	✓	✓	✓	✓
Quintus	✓	✓	✓	✓
Scrawl.js	✓	✓	✓	✓

1 Fabric.js-verkkosivustolta on ladattavissa Sprite sheet -animaatiot lisäävä laajennus.

2 Paper.js-kirjastossa on yksinkertaista toteuttaa Sprite sheet -animaatiot peruskomentojen avulla.

Taulukko 6.1: Canvas-kirjastojen piirto-ominaisuudet 1/2.

kirjasto	**tween-animointi**	**graafisten elementtien törmäyksen tarkistus**	**tekstielementit**	**hiiren ja kosketusnäytön tuki**
CraftyJS	✓	✓	✓	hiiri, kosketusnäyttö, multitouch-toiminnallisuus
Easel.js	✓	✓	✓	hiiri, kosketusnäyttö, multitouch-toiminnallisuus
Fabric.js	✓	✓	✓	hiiri, kosketysnäyttö, multitouch-toiminnallisuus
Konva.js	✓	✓	✓	hiiri, kosketusnäyttö, multitouch-toiminnallisuus
Melon.js	✓	✓	✓	hiiri, kosketusnäyttö
oCanvas.js	✓	ei	✓	hiiri, kosketusnäyttö
Paper.js	✓	✓	✓	hiiri, kosketusnäyttö
Pixi.js	✓	✓	✓	hiiri, kosketusnäyttö, multitouch
Quintus	✓	✓	✓	hiiri, kosketusnäyttö
Scrawl.js	✓	✓	✓	hiiri, kosketusnäyttö

Taulukko 6.2: Canvas-kirjastojen piirto-ominaisuudet 2/2.

kirjasto	tehokkuusoptimointi	WebGL-tuki
CraftyJS	-	WebGL-tuki osalle elementeistä
Easel.js	Cache as Bitmap, Canvas-layerointi	✓
Fabric.js	Canvas-piirron estäminen elementtien poistamisessa ja lisäämisessä	-
Konva.js	Cache as Bitmap, Canvas-layerointi	-
Melon.js	-	✓
oCanvas.js	-	-
Paper.js	-	-
Pixi.js	Cache as Bitmap	✓
Quintus	-	-
Scrawl.js	Canvas-layerointi	-

Taulukko 6.3: Canvas-kirjastojen käyttöliittymäominaisuudet.

6.2 Kirjastojen versionhallintatiedot

kirjasto	Commits	Contributors	Pull requests closed	Pull requests open	projektin aloituspäivä
CraftyJS	1839	101	598	16	31.10.2010
Easel.js	1389	35	146	20	23.1.2011
Fabric.js	3372	128	883	27	6.6.2010
Konva.js	1771	63	28	1	26.2.2012
Melon.js	3970	41	173	0	10.4.2011
oCanvas.js	369	7	21	1	9.1.2011
Paper.js	6552	58	177	12	6.2.2011
Pixi.js	2871	162	899	20	20.1.2013
Quintus	256	33	63	7	29.7.2012
Scrawl.js	297	1	3	0	1.9.2013

Taulukko 6.4: Canvas-kirjastojen versionhallintatietoja 1/2, 27.5.2016.

kirjasto	Issues closed	Issues open	Watchers	Forks
CraftyJS	372	48	136	477
Easel.js	494	104	394	1483
Fabric.js	1903	201	328	1052
Konva.js	108	15	63	78
Melon.js	516	103	123	380
oCanvas.js	79	25	28	55
Paper.js	778	90	360	661
Pixi.js	1445	243	650	2028
Quintus	69	43	135	422
Scrawl.js	0	0	5	4

Taulukko 6.5: Canvas-kirjastojen versionhallintatietoja 2/2, 27.5.2016.

kirjasto	Commits	Contributors
CraftyJS	111	10
Easel.js	59	8
Fabric.js	325	12
Konva.js	116	8
Melon.js	590	11
oCanvas.js	21	2
Paper.js	956	11
Pixi.js	188	26
Quintus	4	3
Scrawl.js	102	1

Taulukko 6.6: Canvas-kirjastojen versionhallintatietoja 27.5.2015–27.5.2016.

6.3 Hylätyt kirjastot

Teknisten ominaisuuksien ja kehitysprojektin versionhallintatietojen perusteella tutkimuksessa hylättiin kirjastot oCanvas.js, Quintus ja Scrawl.js.

- oCanvas.js-kirjasto hylättiin, koska se ei sisällä graafisten elementtien törmäyksen tarkistustoiminnallisuuksia.
- Quintus-kirjasto hylättiin viimeisen vuoden tapahtuneen kehitystyön puutteen vuoksi. Quintus-kirjastoon on tehty muutoksia vain neljä kertaa viimeisen vuoden aikana. Tästä syystä kirjaston kehitysprojekti ei vaikuta aktiiviselta.

- Scrawl.js-kirjasto hylättiin, koska sen kehitystyötä on tehnyt vain yksi henkilö. Kirjaston versionhallintaan ei myöskään ole tullut yhtään ominaisuuksia käsittelevää kehitysehdotusta. Koska kirjaston kehitystyö on vain yhden henkilön varassa, on suuri riski, että kehitystyö loppuu jossain vaiheessa ilman ennakkovaroitusta.

6.4 Ominaisuuksien tarkastelun läpäisseet kirjastot

Taulukossa 6.7 on listattu ne ohjelmointikirjastot, jotka läpäisivät esikarsinnan, ja jotka otetaan seuraavaksi lähempään tarkasteluun.

kirjasto	**kotisivu**
CraftyJS	`http://craftyjs.com/`
Easel.js	`http://www.createjs.com/easeljs/`
Fabric.js	`http://fabricjs.com/`
Konva.js	`https://konvajs.github.io/`
Melon.js	`http://melonjs.org/`
Paper.js	`http://paperjs.org/`
Pixi.js	`http://www.pixijs.com/`

Taulukko 6.7: Ominaisuuksien tarkastelun läpäisseet HTML5 Canvas -ohjelmointikirjastot.

Luku 7

Kirjastojen tehokkuusvertailu

Kirjastojen tehokkuusvertailu tehdään versioilla, jotka on julkaistu kirjastojen virallisilla verkkosivuilla. Taulukossa 7.1 on listattu tehokkuusvertailuun valitut kirjastot ja niiden versiot.

kirjasto	**versionumero**	**julkaisuaika**
CraftyJS	0.7.1	27.2.2016
Easel.js	0.8.2	4.12.2015
Fabric.js	1.6.2	16.5.2016
Konva.js	0.13.0	16.6.2016
Melon.js	3.1.0	16.5.2016
Paper.js	0.9.25	25.10.2015
Pixi.js	3.0.11	3.13.2016

Taulukko 7.1: Tehokkuustestauksessa käytettyjen HTML5 Canvas -ohjelmointikirjastojen versiot.

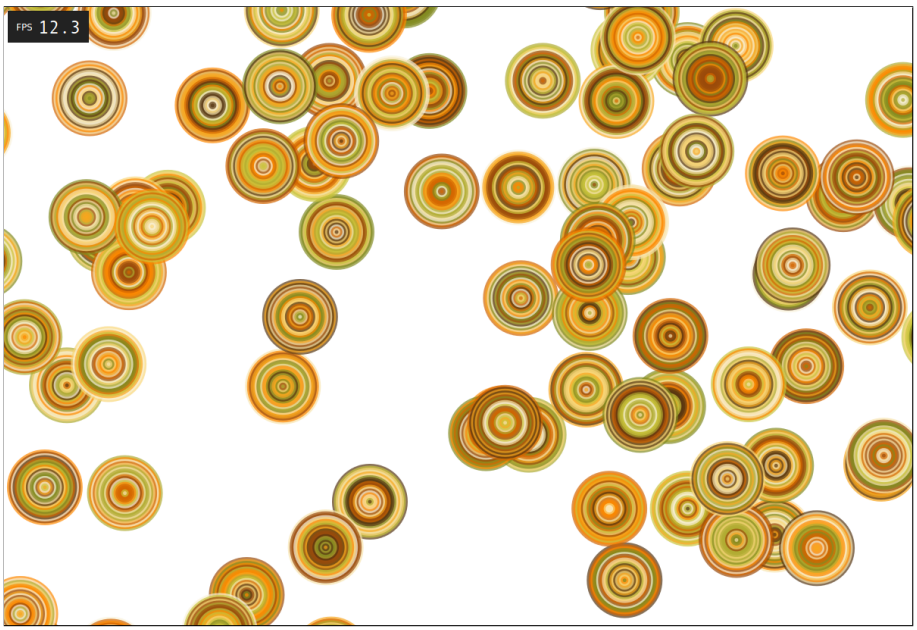
7.1 Testiohjelmat

Tehokkuustestauksessa testataan kirjastojen tehokkuutta piirtää ruudulle vektori- ja bittikarttagrafiikkaa. Testausta varten jokaiselle kirjastolle on ohjelmoitu samanlainen testiohjelma vektorigrafiikka- ja bittikarttagrafiikkatestille. Windows 10 -ympäristössä testiohjelmat toteutettiin resoluutiossa 1200 x 900. Ipad-tabletissa resoluutiona käytettiin Retina-näytön natiiviaresoluutiota (2048 x 1536), jotta Ipadin selaimessa tekemä skaalaus ei vaikuttaisi kirjastojen toimintaan.

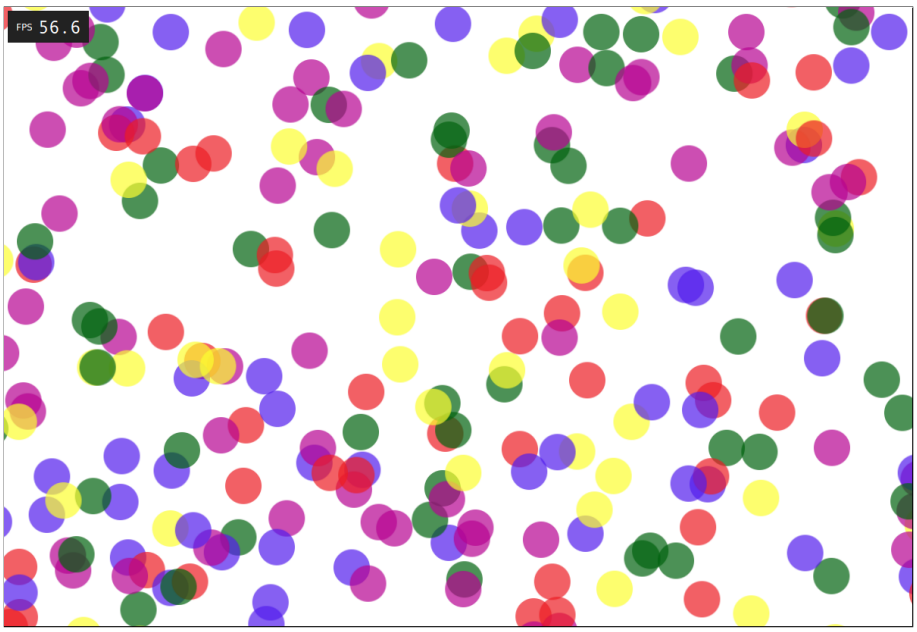
Vektorigrafiikan piirtonopeutta testaavassa ohjelmassa liikutetaan vek-

torigrafiikalla piirrettäviä elementtejä. Kun testiohjelma käynnistyy, se luo testattavan määrän graafisia objekteja. Jokaisen objektin sisälle piirretään 20 satunnaisen väristä sisäkkäistä keskitettyä ympyrää. Ensimmäisen ympyrän leveys on 100 pikseliä ja jokainen seuraava ympyrä on aina $\frac{1}{20}$ edellistä pienempi. Jokainen piirrettävä ympyrä on 30 % läpinäkyvä. Objekteille annetaan alustuksessa satunnainen suunta ja nopeus. Ruudunpäivityksen laskevassa funktiossa lasketaan objekteille aina uudet positiot ruudulla perustuen niiden alustuksessa määritettyyn nopeuteen ja suuntaan. Graafisen objektin siirtyessä Canvas-elementin reunan ulkopuolelle, objekti siirretään vastakkaiselle puolelle elementtiä.

Bittikarttagrafiikan piirtonopeutta testaavassa ohjelmassa liikutetaan Sprite sheet -tekniikalla toteutettuja animoitua objekteja. Kun testiohjelma käynnistyy, se luo testattavan määrän graafisia objekteja. Jokaiseen objektiin liitetään viisi animaatiokuvaa sisältävä kuvasarja. Jokaisessa objektissa animoidaan tätä viiden kuvan sarjaa siten, että ruudunpäivityksessä animaatio siirtyy aina yhden kuvan eteenpäin. Samoin kuin vektorigrafiikkatestissä objektit on asetettu 30 % läpinäkyviksi, ja niille annetaan alustuksessa satunnainen suunta ja nopeus. Ruudunpäivityksen laskevassa funktiossa lasketaan objekteille aina uudet positiot ruudulla perustuen niiden nopeuteen ja suuntaan. Graafisen objektin siirtyessä Canvas-elementin reunan ulkopuolelle, objekti siirretään vastakkaiselle puolelle elementtiä.



Kuva 7.1: Vektorigrafiikan piirtotesti Windows 10 -ympäristössä.



Kuva 7.2: Bittikarttagrafiikan piirtotesti Windows 10 -ympäristössä.

Kaikki testiohjelmat toimivat yhden ruudunpäivitysfunktion kautta. Ruudunpäivitysfunktio laskee graafiset muutokset, piirtää grafiikan uudestaan ruudulle ja kutsuu tämän jälkeen itseään selaimen window.requestAnimationFrame-funktion avulla. Window.requestAnimationFrame ajaa sille annetun funktion, ennen kuin selain päivittää ruudun uudestaan. Tällöin grafiikan uudelleenpiirto tehdään maksimissaan niin usein, kuin laitteen näytön ruudunpäivitys tapahtuu. Tällöin ei kuluteta resursseja grafiikan päivittämiseen ilman, että se näkyy ruudulla. Testeissä käytetyssä Ipad-tabletissa Safari-selaimella maksimiruudunpäivitysnopeus on 30 kertaa sekunnissa ja Windows 10 tietokoneessa Chrome-selaimella 50 kertaa sekunnissa.

7.1.1 Ruudunpäivityksen mittaaminen

Testiohjelmien ruudunpäivitysnopeuden seuraaminen tehdään FPSMeter[1]-ohjelman avulla. Jokaisen testiohjelman käynnistyessä käynnistetään FPSMeter-ohjelma, joka näyttää ruudunpäivitysnopeuden. FPSMeter-ohjelmaa kutsutaan ruudun uudelleen piirtävän funktion alussa. Tällä tavoin FPSMeter mittaa, kuinka monta kertaa ruutu on piirretty sekunnin

[1]http://darsa.in/fpsmeter/

aikana. FPSMeter-ohjelmassa on asetettu smoothing-parametrin arvoksi 20, jolloin ruudunpäivitysnopeuden esityksestä tasoitetaan hetkittäiset pienet muutokset.

7.1.2 Testiautomaatio

Testausta varten tutkimuksessa ohjelmoitiin yksittäisten testien lisäksi automaattinen testausjärjestelmä, joka käy ennalta määritellyn testijoukon läpi testilaitteella. Automaattinen testausjärjestelmä luo käynnistyessään testijoukon URL-linkit testityypin ja elementtimäärän mukaan sekä tallentaa ne selaimen localstorage-tallennustilaan. Tämän jälkeen testausjärjestelmä käy yksitellen testijoukon läpi ja tallentaa testin tulokset palvelimelle. Testiautomaatti käy testitapaukset läpi seuraavalla tavalla:

1. Käynnistetään testiautomaatti.
2. Käynnistetään yksittäinen testiohjelma.
3. Odotetaan kymmenen sekuntia, että ruudunpäivityslaskuri on tasaantunut.
4. Otetaan sekunnin välein näyte ruudunpäivityslaskurin arvosta.
5. Kymmenen näytteen jälkeen lasketaan tallennettujen näytteiden keskiarvo ja tallennetaan se selaimen muistiin.
6. Siirrytään seuraavaan testitapaukseen.

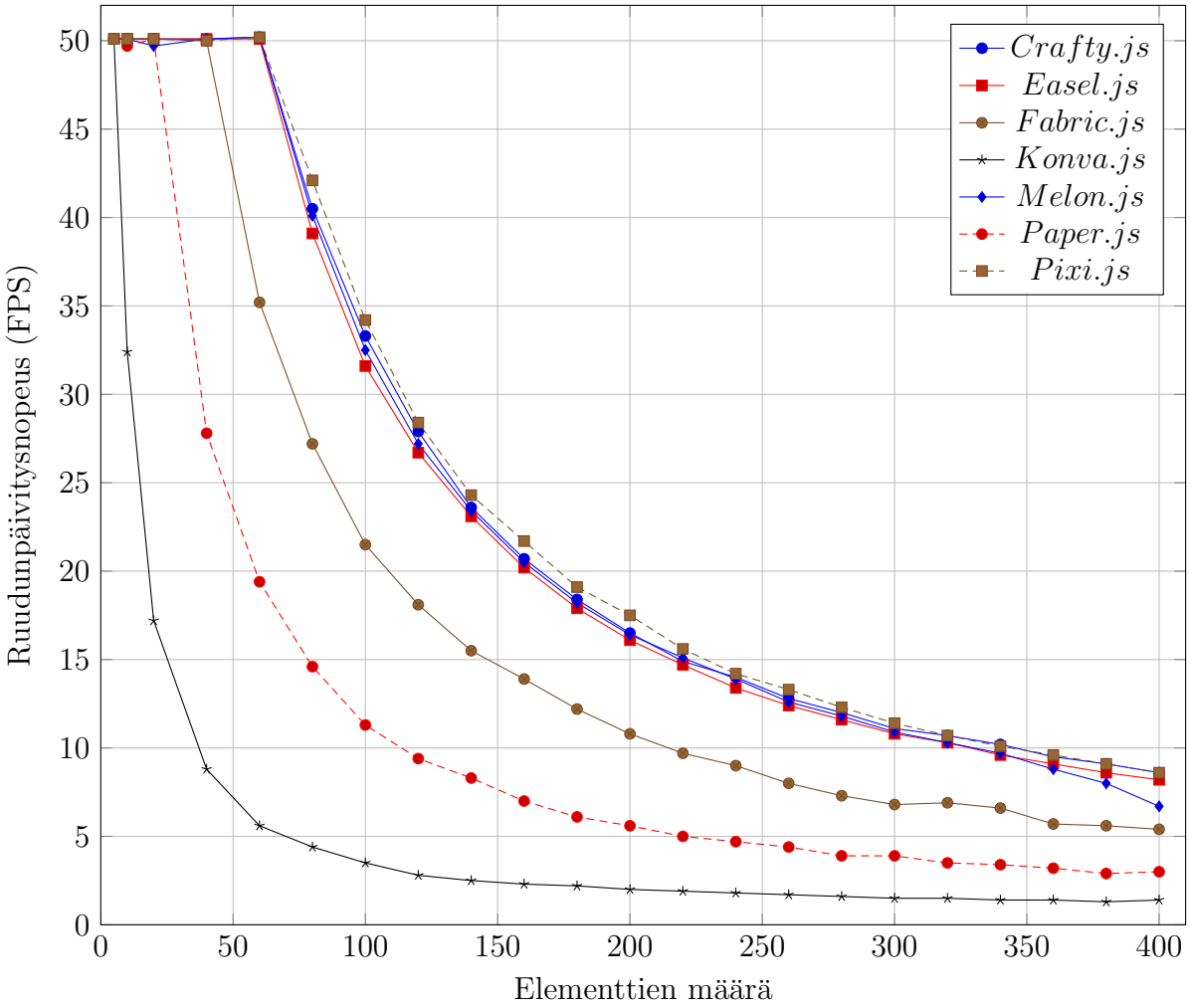
7.2 Testiympäristöt

Tehokkuustestaukset suoritetaan sekä mobiililaitteessa että tavallisessa pöytätietokoneessa. Mobiililaitteena käytetään Ipad-tablettia ja pöytäkoneena Windows 10 -käyttöjärjestelmällä varustettua tietokonetta. Laitteiden tarkemmat tiedot on esitetty taulukossa 7.2.

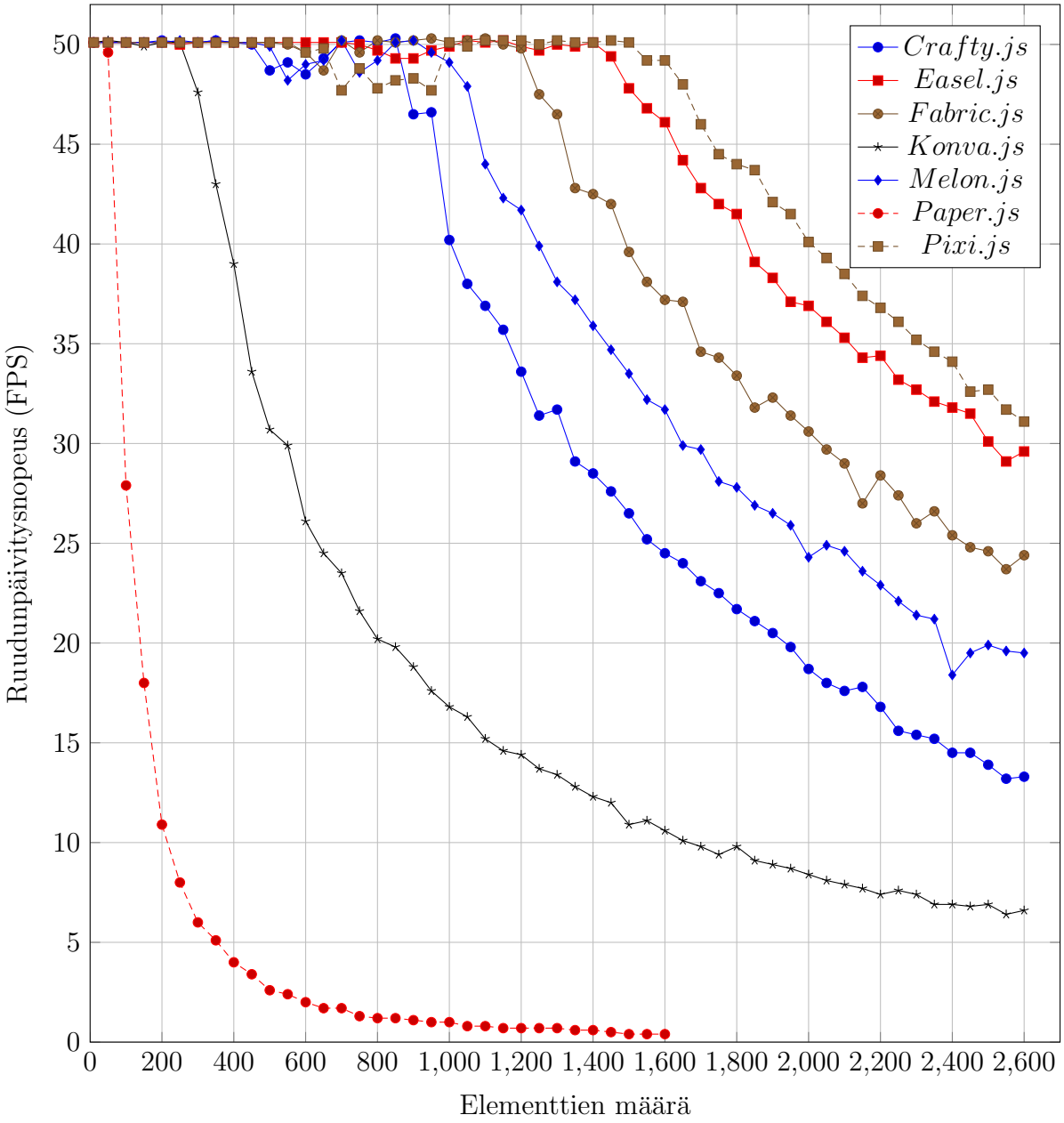
Testiympäristöt	
laitemalli	Ipad 3rd gen. retina 32 Gt
käyttöjärjestelmä	iOS 8.4.1
prosessori	1 GHz dual-core ARM Cortex-A9
näytönohjain	Quad-core PowerVR SGX543MP4
muisti	1 Gt
näyttö	2048 x 1536
selain	Safari
laitemalli	Lenovo e531
käyttöjärjestelmä	Windows 10
prosessori	Intel i7-3632QM
näytönohjain	Nvidia GeForce GT 740M
muisti	12 Gt
näyttö	1920 x 1200
selain	Chrome 51.0

Taulukko 7.2: Tehokkuustestauksessa käytetyt testiympäristöt.

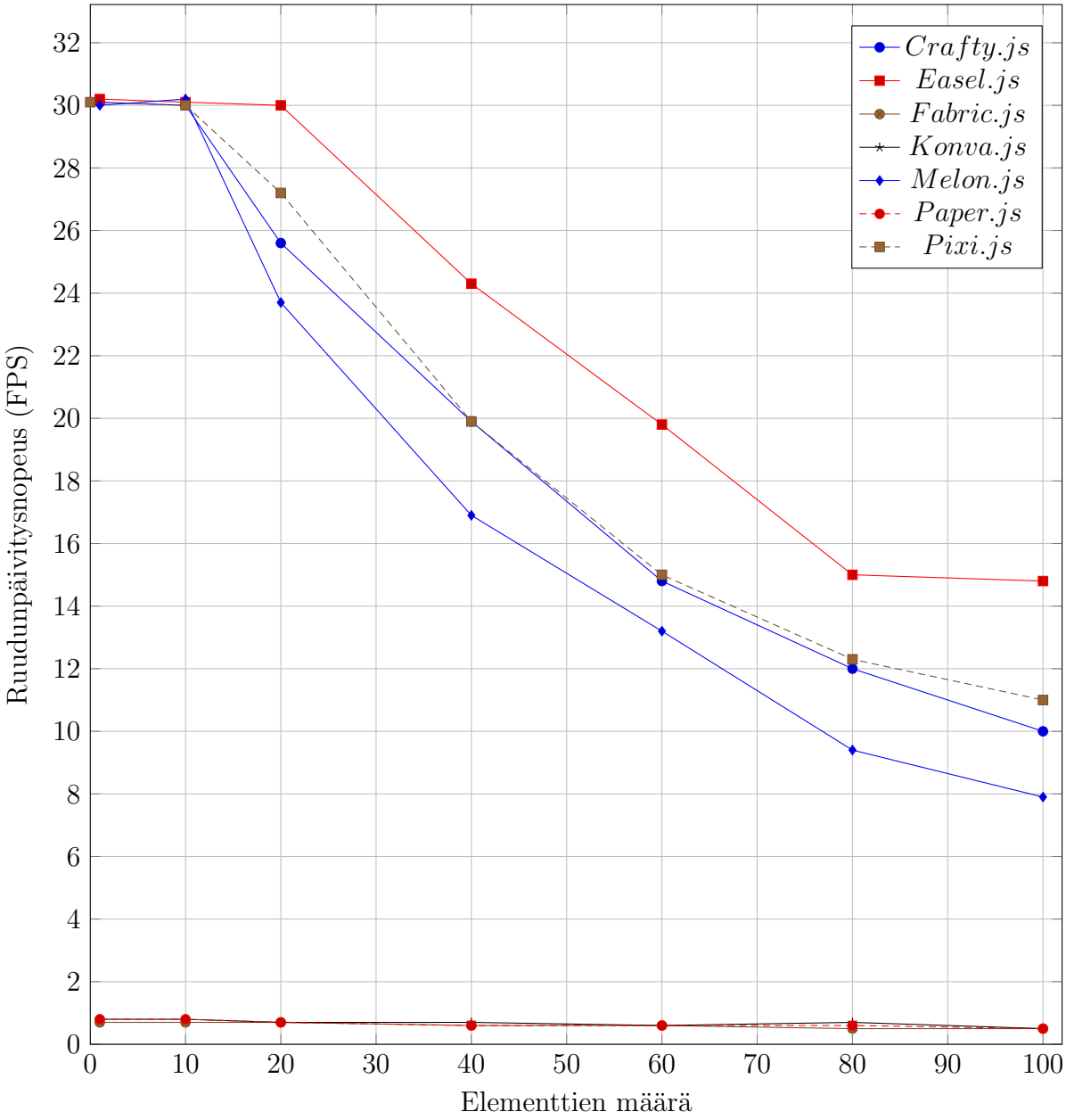
7.3 Piirtonopeustestit



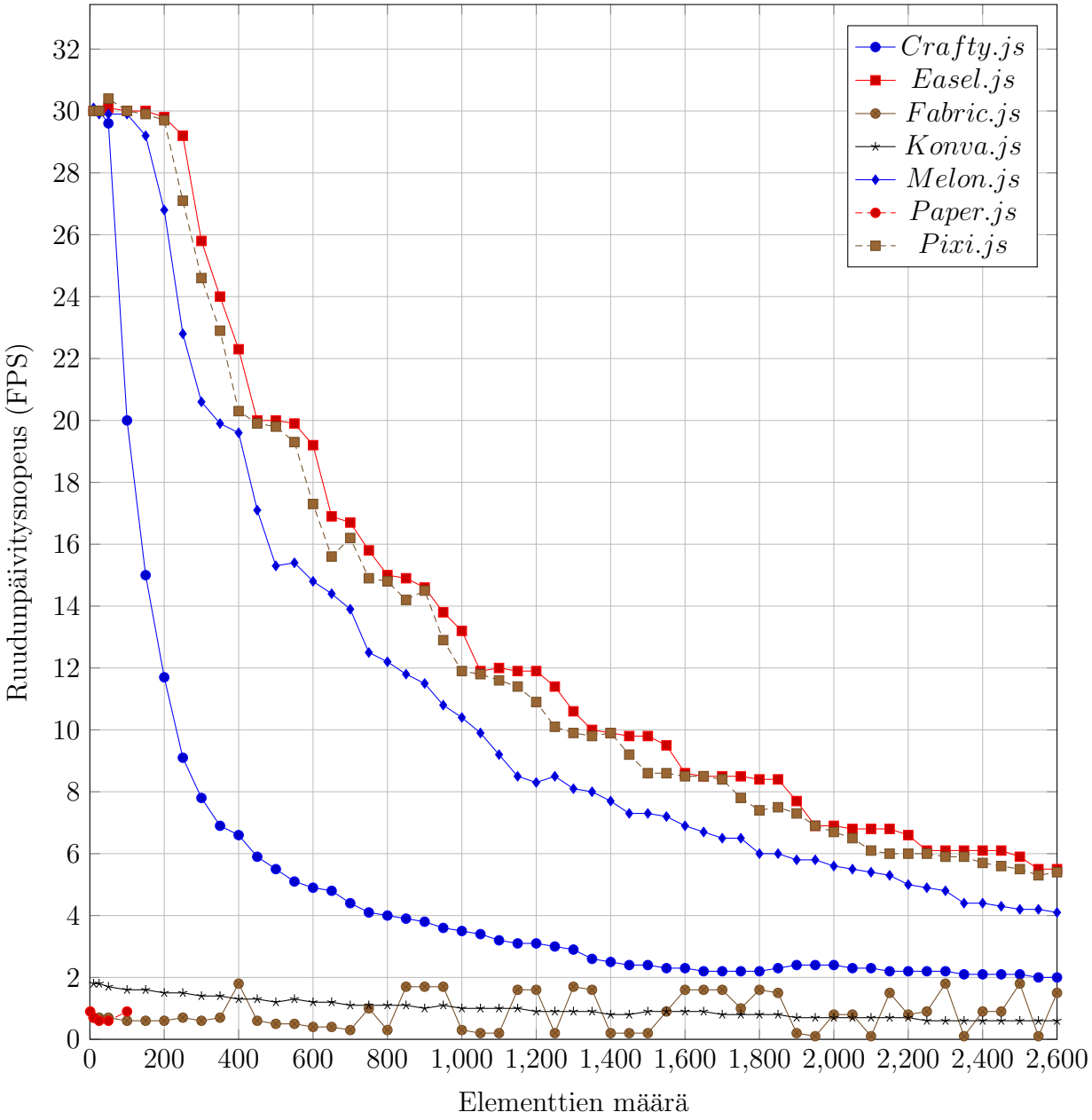
Kuva 7.3: Vektorigrafiikkatestin tulokset Windows 10 -ympäristössä.



Kuva 7.4: Bittikarttagrafiikkatestin tulokset Windows 10 -ympäristössä.



Kuva 7.5: Vektorigrafiikkatestin tulokset Ipad-laitteella.



Kuva 7.6: Bittikarttagrafiikkatestin tulokset Ipad-laitteella.

7.4 Muistinkäyttötestit

Kirjastojen muistinkäyttötestaus tehtiin Windows 10 -ympäristössä. Testiohjelmien muistinkäyttöä seurattiin Chrome-selaimen profilointityökaluilla,

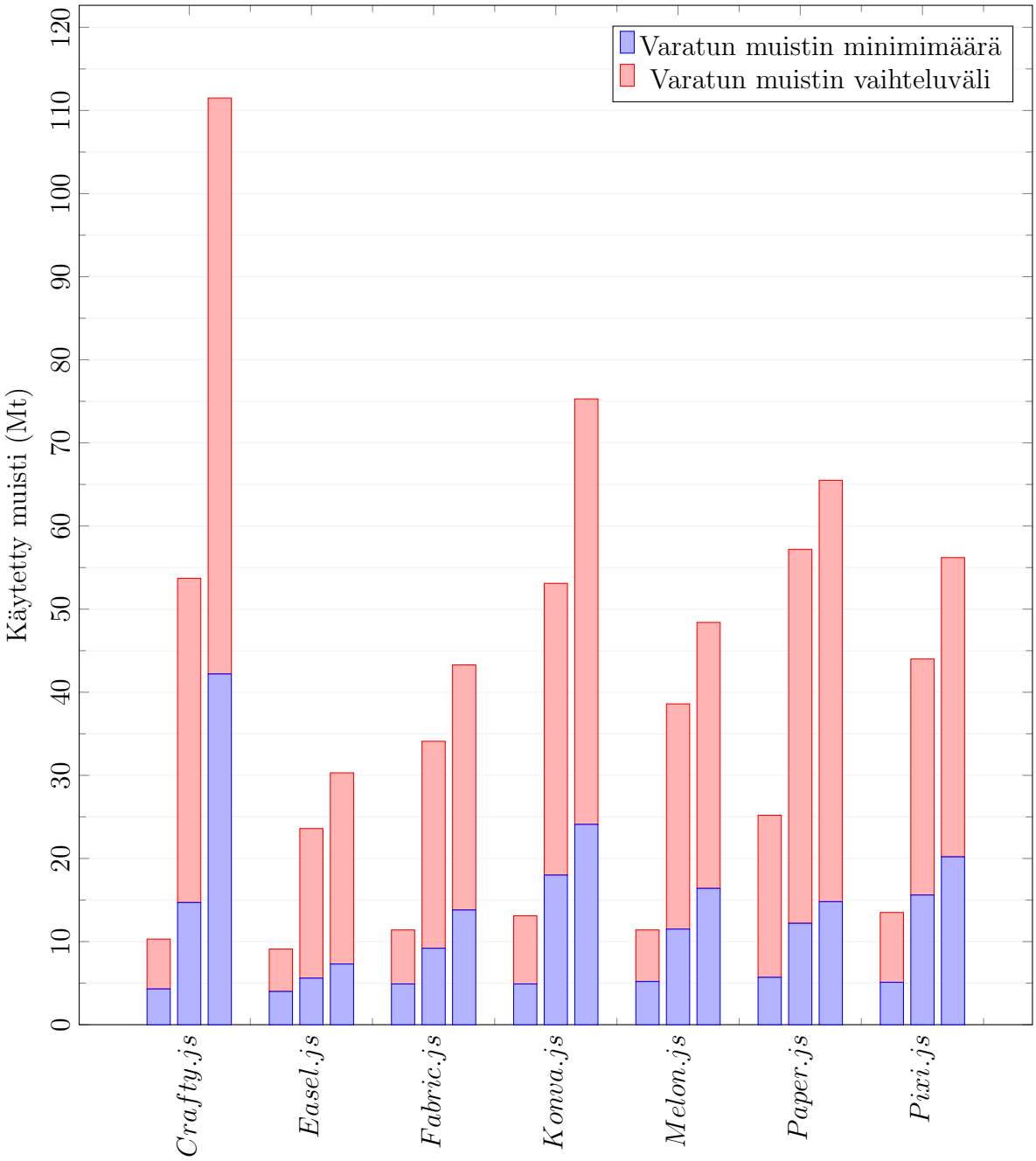
joilla seurattiin testiohjelman ajonaikaista allokoidun muistin määrää ja sen vaihtelua. Muistinkäyttötestauksessa seurattiin kirjastoja myös muistivuotojen osalta. Niitä ei kuitenkaan havaittu.

Testiohjelmat on ohjelmoitu siten, että ne kuluttavat mahdollisimman vähän resursseja kirjaston oman toiminnan ulkopuolella. Yksi suurimmista reaaliaikaista grafiikkaa piirtävän JavaScript-ohjelman hidasteista on JavaScript-tulkin roskankeruualgoritmin (Garbage Collector) ajaminen ohjelman suorituksen aikana.

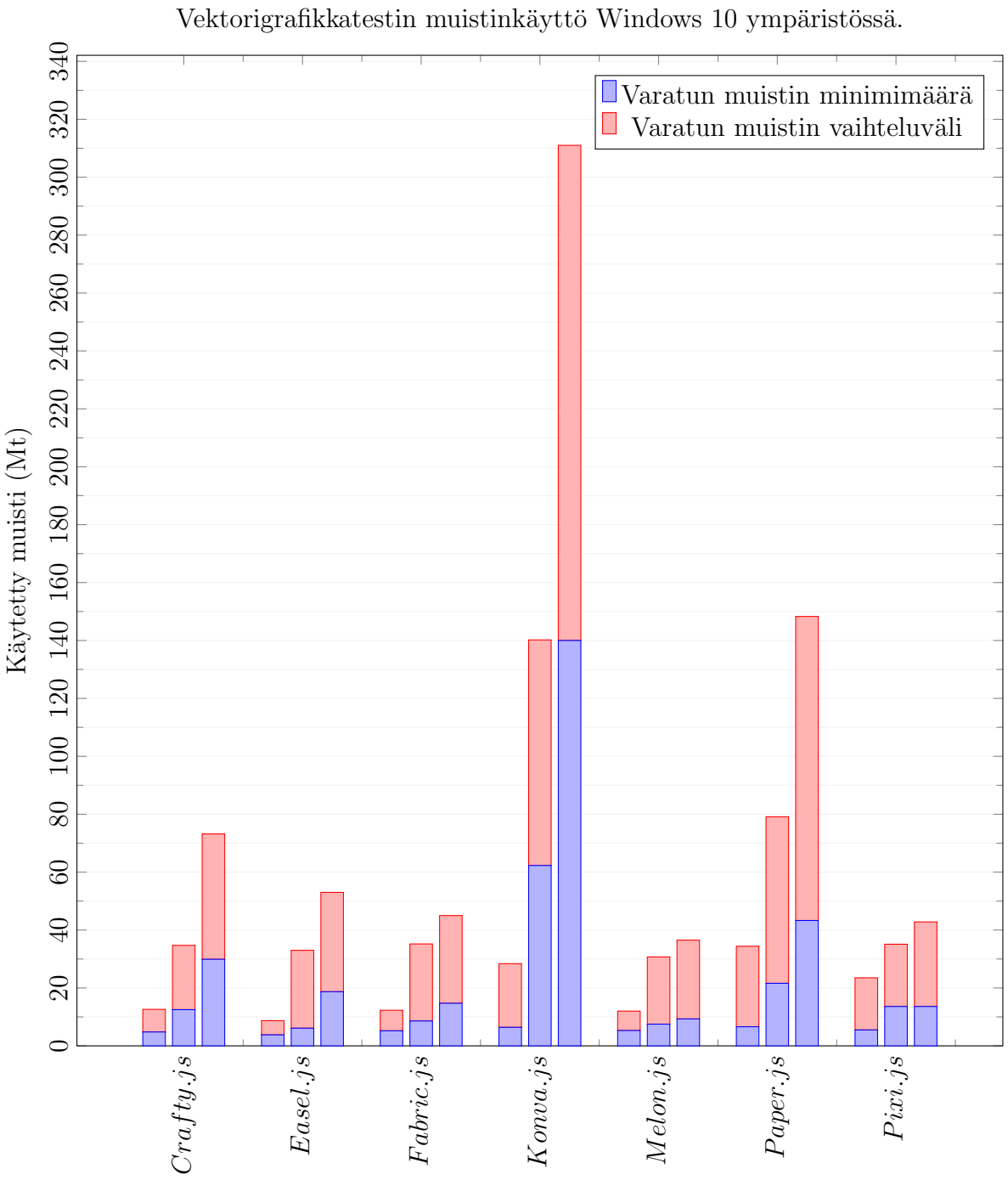
Roskankeruualgoritmin tarkoitus on vapauttaa ohjelman varaamaa muistia niiden resurssien osalta, joita ei enää tarvita. Yksittäinen roskankeruualgoritmin ajokerta saattaa kestää useita millisekunteja. Jos ruudunpäivitysnopeus on 50 kuvaa sekunnissa, jää yhden kuvan piirtämiseen aikaa 20 millisekuntia. Tällöin roskankeruualgoritmin käynnistyminen vaikuttaa selvästi piirtotehokkuuteen. Testiohjelmien rakenne suunniteltu siten, että ruudunpäivityksen suorittava piirtofunktio ei varaa uusia resursseja tai suorita uusia alifunktioita. Tällä tavoin minimoidaan testiohjelman vaikutus roskankeruualgoritmin käynnistymiseen.

Kirjastojen vaikutus roskankeruualgoritmin toimintaan nähdään testiohjelmien muistinkäyttöä esittävissä kuvaajissa 7.7 ja 7.8.

Bittikarttagrafikkatestin muistinkäyttö Windows 10 ympäristössä.



Kuva 7.7: Kuvaaja esittää jokaisen kirjaston käyttämän muistin määrän 10, 1000 ja 2000 elementin testissä.



Kuva 7.8: Kuvaaja esittää jokaisen kirjaston käyttämän muistin 10, 400 ja 1000 elementin testissä.

Luku 8

Analyysi

8.1 Crafty.js

Crafty.js on osin peliohjelmointiin erikoistunut kirjasto. Se täyttää tutkimuksessa vaaditut ominaisuudet ja tukee myös WebGL-piirtoa osalle elementeistä. Crafty.js soveltuu myös yleiskäyttöiseen Canvas-ohjelmointiin. Kirjaston kehitysprojekti toimii aktiivisesti GitHub-versionhallinnassa.

Tehokkuustesteissä Crafty.js ylsi vektorigrafiikan piirrossa lähes kärkitulokseen yltäneiden kirjastojen tasolle, mutta bittikarttagrafiikkatestissä se oli selvästi hitaampi. Vektorigrafiikkatestissä Crafty.js oli Windows 10 -ympäristössä neljän nopeimman kirjaston joukossa ja Ipad-ympäristössä jaetulla toisella sijalla. Sen sijaan bittikarttagrafiikkatestissä piirtonopeus jäi selvästi heikommaksi sekä Windows 10- että Ipad-ympäristöissä.

Muistinkäyttöä tarkasteltaessa juuri bittikarttagrafiikkatestissä Crafty.js varasi muistia huomattavasti enemmän kuin muut kirjastot. Varatun muistin käyttö nousi nopeammin kuin muissa kirjastoissa elementtien määrän lisääntyessä. Vektorigrafiikkatestissä kirjasto käytti muistia maltillisemmin, eikä samanlaista muistinkäytön kasvamista tapahtunut.

8.2 Easel.js

Easel.js-kirjasto kuuluu Adoben tukemaan Create.js-nimiseen HTML5-ohjelmointikirjastojen kokoelmaan, joka on tarkoitettu yleiskäyttöisten Canvas-sisältöjen koostamiseen. Ominaisuuksiltaan kirjasto täyttää tutkimuksessa asetetut vaatimukset ja soveltuu hyvin yleiskirjastoksi erilaisiin laitteisiin suunnattujen sisältöjen kehittämiseen. GitHub-versionhallinnassa kirjastolla on aktiivinen kehittäjäjoukko.

Tehokkuustestauksessa Easel.js jakoi parhaan tuloksen Pixi.js-kirjaston

kanssa. Window 10 -ympäristössä Easel.js-kirjasto saavutti vektorigrafiikkatestissä lähes saman tuloksen kuin testin nopein Pixi.js-kirjasto ja sijoittui testin neljän nopeimman kirjaston joukkoon. Bittikarttagrafiikkatestissä Easel.js-kirjasto oli noin 5–10 % Pixi.js-kirjastoa hitaampi, kun piirrettävien elementtien määrä oli yli 1400 kappaletta. Ipad-ympäristössä Easel.js oli molemmissa testeissä tutkimusjoukon nopein. Vektorigrafiikkatestissä toiseksi parhaan tuloksen saavuttanut Pixi.js jäi noin 15–30 % Easel.js-kirjaston nopeudesta, kun elementtien määrä nousi yli 10 kappaleeseen. Bittikarttagrafiikkatestissä Easel.js oli Pixi.js-kirjastoa nopeampi noin 5–10 %:n marginaalilla osassa testitapauksia; osassa testitilanteita tulos oli lähes sama.

Bittikarttagrafiikkatestissä Easel.js varasi vähiten muistia testattujen kirjastojen joukosta. Vektorigrafiikkatestissä Easel.js varasin hieman enemmän muistia kuin muistinkäytöltään tehokkaimmat kirjastot. Molemmissa testeissä varatun muistin määrän kasvu oli maltillista.

8.3 Fabric.js

Fabric.js on yleiskäyttöiseen Canvas-ohjelmointiin tarkoitettu kirjasto. Se täyttää tutkimuksessa vaaditut ominaisuudet. Sprite sheet -animaatioiden osalta kirjastossa ei ole valmista toiminnallisuutta, mutta siihen on satavilla virallinen laajennus, jolla kirjaston toiminta voidaan kattaa sisältämään Sprite sheet -animaatiot. Kirjastolla on GitHub-versionhallintajärjestelmässä aktiivisesti toimiva kehitysprojekti.

Tehokkuustestauksessa Windows 10 -ympäristössä Fabric.js-kirjaston testitulokset jäivät molemmissa testeissä jonkin verran nopeimpien kirjastojen tuloksista. Ipad-ympäristössä Fabric.js-kirjasto ei saavuttanut kunnollista tulosta, vaan sen piirtonopeus oli yksi ruutu sekunnissa riippumatta elementtien määrästä. Tämä saattoi johtua Ipad-laitteella tehdyissä testissä käytetystä korkeasta resoluutiosta.

Muistinkäytössä Fabric.js sijoittui molemmissa testeissä pienimmän määrän muistia varanneiden kirjastojen joukkoon.

8.4 Konva.js

Konva.js on vanhentuneen Kinetic.js-kirjaston lähdekoodista edelleen kehitetty versio. Kirjasto on tarkoitettu yleisluontoiseen Canvas-elementtiä käyttävien ohjelmien kehitykseen. Konva.js sisältää tutkimuksessa tarkastelluista toiminnallisuuksista perusominaisuudet ja piirto-optimointeja. WebGL-tukea ei ole tarjolla. Konva.js-kehitysprojekti toimii aktiivisesti

GitHub-versionhallinnassa.

Tehokkuuden osalta Windows 10 -ympäristössä Konva.js suoriutui vektorigrafiikkatestissä kaikista heikoiten. Bittikarttagrafiikkatestissä Konva.js suoriutui toiseksi heikoiten. Piirrettävien elementtien määrän noustessa yli 250 piirtonopeudessa tapahtui nopea pudotus. Ipad-ympäristössä Konva.js suoriutui heikosti molemmissa testeissä. Vektorigrafiikkatestissä Konva.js:n piirtonopeus oli yksi ruudunpäivitys sekunnissa riippumatta elementtien määrästä. Bittikarttagrafiikkatestissä elementtien määrän kasvu vaikutti tulokseen, mutta tällöinkin nopein piirtonopeus on kaksi ruudunpäivitystä sekunnissa. Todennäköisesti Ipad-laitteen Retina-näytön suuri resoluutio vaikutti testiin.

Konva.js-kirjaston käyttämän muistin määrä oli vektorigrafiikkatestissä selvästi muita korkeampi. Varatun muistin määrän nousi huomattavasti suuremmalla nopeudella kuin muissa kirjastoissa, kun elementtien määrä kasvoi. Bittikarttagrafiikkatestissä muistinkäyttö oli elementtien lukumäärän noustessa testin suurimpien joukossa, mutta varatun muistin määrä ei noussut yhtä nopeasti kuin vektorigrafiikkatestissä.

8.5 Melon.js

Melon.js on tarkoitettu muun muassa nopeaan pelikehitykseen. Sen ominaisuudet painottuvat enemmän käyttöliittymäelementtien ja peliobjektien toiminnallisuuksiin kuin grafiikan piirtämiseen. Melon.js sisältää kaikki tutkimuksessa asetetut perusominaisuudet grafiikan piirtämiseen ja lisäksi WebGL-tuen. Melon.js-kehitysprojekti toimii aktiivisesti GitHub-versionhallinnassa. Kirjaston dokumentaatio ei ole yhtä kattava kuin parhaiten dokumentoiduissa kirjastoissa, mutta riittää sen toimintaan perehtymiseen.

Tehokkuustestauksessa Windows 10 -ympäristössä Melon.js suoriutui vektorigrafiikkatestissä neljän nopeimman kirjaston kanssa samaan luokkaan. Bittikarttagrafiikkatestissä Melon.js suoriutui selvästi heikommin kuin parhaaseen tulokseen yltänyt Pixi.js-kirjasto. Kun piirrettävien elementtien määrä nousi yli 1000 kappaleen, Melon.js-kirjaston ruudunpäivitysnopeus alkoi jäädä testin kärkituloksista. Ipad-tabletilla Melon.js suoriutui vektorigrafiikkatestissä noin 30–40 % hitaammin kuin testin nopein Easel.js-kirjasto. Bittikarttagrafiikkatestissä ero oli 15–25 % nopeimpaan Easel.js kirjastoon.

Vektorigrafiikkatestissä Melon.js-kirjaston muistinkäyttö edusti keskivertoa testatuista kirjastoista. Vektorigrafiikkatestissä Melon.js oli vähiten muistia varaavien kirjastojen joukossa. Molemmissa testeissä Melon.js varaaman muistin määrä kasvoi maltillisesti elementtien määrän kasvaessa.

8.6 Paper.js

Paper.js on erityisesti vektorigrafiikan piirtämiseen tarkoitettu kirjasto. Se sisältää heikommat ominaisuudet bittikarttagrafiikan ohjelmoimiseen, mutta se täyttää silti tutkimuksessa asetetut vaatimukset. Sprite sheet -animaatioiden tekemiseen ei löydy kirjastosta suoraa toiminnallisuutta, mutta virallisilla verkkosivuilla on Sprite sheet -animaatioiden tekemiseen tarkoitettu ohjelmakoodiesimerkki. Paper.js-kirjastolla on aktiivinen kehitysprojekti GitHub-versionhallintasivustolla.

Tehokkuustestauksessa Paper.js oli hitaimpien joukossa Windows 10 -ympäristön vektorigrafiikkavertailussa. Bittikarttagrafiikkatestissä Paper.js oli testin hitain. Kun elementtien määrä ylitti 1600 kappaletta, testaus lopetettiin, koska testiohjelman käynnistäminen kesti useita minuutteja. Ipad-ympäristössä vektorigrafiikkatestissä Paper.js oli testin kolmen hitaimman kirjaston joukossa, jotka eivät suorittaneet yli yhtä ruudunpäivitystä sekunnissa riippumatta elementtien määrästä. Bittikarttatestissä testaus jouduttiin lopettamaan 100 elementin jälkeen, koska testiohjelman suoritus ei enää edennyt.

Muistinkäytössä Paper.js oli eniten muistia varanneiden kirjastojen joukossa. Varsinkin vektorigrafiikkatestissä Paper.js varasi selvästi enemmän muistia suurilla elementtimäärillä kuin muut kirjastot lukuun ottamatta Konva.js kirjastoa.

8.7 Pixi.js

Pixi.js on yleiseen Canvas-sisältöjen kehittämiseen tarkoitettu kirjasto. Se täyttää tutkimuksessa asetetut toiminnallisuusvaatimukset ja sisältää myös nopeuteen liittyviä optimointeja. Pixi.js käyttää oletuksena WebGL-tekniikkaa grafiikan piirtämiseen, mutta se sisältätää myös Canvas-tuen vanhemmille laitteille. Pixi.js-kirjastolla on aktiivinen kehitysprojekti GitHub-versionhallinnassa. Pixi.js kirjastoa käytetään myös piirtokirjastona HTML5-pelikirjastoissa, kuten suositussa Phaser.io-kirjastossa.

Tehokkuustestauksessa Pixi.js jakoi kärkisijan Easel.js-kirjaston kanssa. Windows 10 -ympäristössä Pixi.js oli Easel.js-kirjastoa hieman nopeampi molemmissa testeissä, joista bittikarttagrafiikkatestissä eroa oli noin 5–10 %, kun elementtien määrä ylitti 1400 kappaletta. Ipad-ympäristössä vektorigrafiikkatestissä Pixi.js jäi noin 15–30 % Easel.js-kirjaston kärkituloksesta. Bittikarttatestissä nopeus oli molemmissa kirjastoissa elementtien määrän vaihdellessa välillä sama ja välillä Easel.js oli noin 5–10 % nopeampi.

Muistinkäyttössä Pixi.js-kirjasto edusti bittikarttagrafiikkatestissä medi-

aania. Vektorigrafiikkatestissä Pixi.js oli vähiten muistia varaavien kirjastojen joukossa. Molemmissa testeissä Pixi.js varaaman muistin määrä kasvoi maltillisesti elementtien määrän kasvaessa.

Luku 9

Yhteenveto

Tutkimuksen tavoitteena oli löytää yleiskäyttöinen, tehokas ja ominaisuuksiltaan kattava ohjelmointikirjasto HTML5 Canvas -elementin ohjelmoimiseen. Tutkimuksessa löydettiin selviä eroja tutkittujen ohjelmistokirjastojen välillä. Tutkituista ohjelmointikirjastoista parhaiksi valikoituivat Easel.js ja Pixi.js.

Tutkimuksessa tarkasteltiin 42 ohjelmointikirjastoa, jotka on tarkoitettu HTML5 Canvas -elementin graafiseen ohjelmointiin. Kirjastoille asetettujen toiminnallisuutta, graafisia ominaisuuksia ja kirjaston kehitysprojektia koskevien vaatimusten kautta hylättiin 35 kirjastoa. Jäljelle jääneille seitsemälle kirjastolle suoritettiin tehokkuustestaukset, joilla tutkittiin kirjastojen suorituskykyä pöytäkoneessa ja mobiililaitteessa.

Tehokkuustestauksessa testattiin kirjaston suorityskykyä vektori- ja bittikarttagrafiikan piirtämisessä. Suorituskykytesteissä parhaat tulokset saavuttivat Easel.js- ja Pixi.js-kirjastot. Easel.js-kirjasto oli Pixi.js-kirjastoa nopeampi mobiillaitetesteissä. Pöytäkoneella Pixi.js saavutti paremman tuloksen.

Suorituskykytestauksen lisäksi tutkimuksessa tarkasteltiin kirjastojen muistinkäyttöä. Muistinkäytön tarkastelu on tärkeää etenkin mobiililaitteilla, joissa käytettävän muistin määrä on usein huomattavasti rajallisempi. Tutkituista kirjastoista muutama varasi moninkertaisesti muistia muihin kirjastoihin verrattuna. Tehokkuustestauksessa menestyneet Easel.js ja Pixi.js varasivat vektorigrafiikkatestissä maltillisesti muistia. Bittikarttagraffiikkatestissä Pixi.js varasi lähes 50 % enemmän muistia kuin Easel.js suurella elementtimäärällä. Varatun muistin määrä oli molemmissa kuitenkin kohtuullisissa rajoissa.

Lähteet

[1] I Hickson, R Berjon, S Faulkner, T Leithead, E Doyle Navara, and E O'Connor. S. pfeiffer,"html5", w3c recommendation rec-html5-20141028, october 2014. [Viitattu: 21.4.2016]. Saatavissa: `https://www.w3.org/TR/html5/`.

[2] Colin Moock. *Essential ActionScript 3.0*. "O'Reilly Media, Inc.", 2007.

[3] Dave Raggett, Jenny Lam, Ian Alexander, and Michael Kmiec. *Raggett on HTML 4 (2Nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[4] WWW Talk. Archives: Worldwideweb mailing list: Introduction, messages from mon, oct. 28, 1991 to fri, dec. 13, 1991, 1 page, 1991. [Viitattu: 17.4.2016]. Saatavissa: `http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html`.

[5] International Organization for Standardization. *Information Processing: Text and Office Systems: Standard Generalized Markup Language (SGML)*. ISO, 1986.

[6] Ari Luotonen, Henrik Frystyk, and Tim Berners-Lee. Cern httpd. *World Wide Web Consortium Server*, 1996. [Viitattu: 30.5.2016]. Saatavissa: `http://www.w3.org/pub/WWW/Daemon`.

[7] Tim Berners-Lee et al. The worldwideweb browser. *The World Wide Web Consortium (W3C).*, 412, 2006. [Viitattu: 3.5.2016]. Saatavissa: `http://www.w3.org/People/Berners-Lee/WorldWideWeb.html`.

[8] Tim Berners-Lee and Robert Cailliau. Worldwideweb: Proposal for a hypertext project. 1990. [Viitattu: 7.5.2016]. Saatavissa: `https://www.w3.org/Proposal.html`.

[9] Tim Berners-Lee. The original http as defined in 1991. *World Wide Web Consortium (W3C)*, 1991. [Viitattu: 3.5.2016]. Saatavissa: `https://www.w3.org/Protocols/HTTP/AsImplemented.html`.

[10] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. RFC 1945 (Informational), May 1996.

[11] Tim Berners-Lee. Html tags, 1991. [Viitattu: 17.4.2016]. Saatavissa: `http://info.cern.ch/hypertext/WWW/MarkUp/Tags.html`.

[12] T Berners-Lee and D Connolly. Ietf working draft hypertext markup language (html): a representation of textual information and metainformation for retrieval and interchange, 1993. [Viitattu: 4.5.2016]. Saatavissa: `https://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt`.

[13] T. Berners-Lee and D. Connolly. Hypertext Markup Language - 2.0. RFC 1866 (Historic), November 1995. Obsoleted by RFC 2854.

[14] Dave Raggett. Html 3.2 reference specification. *W3C Recommendation, January*, 1997. [Viitattu: 5.5.2016]. Saatavissa: `https://www.w3.org/TR/REC-html32`.

[15] Dave Raggett, Arnaud Le Hors, Ian Jacobs, et al. Hypertext markup language (html) 4.0, 1997. [Viitattu: 10.5.2016]. Saatavissa: `https://www.w3.org/TR/WD-html40/`.

[16] Dave Raggett, Arnaud Le Hors, Ian Jacobs, et al. Html 4.01 specification. *W3C recommendation*, 1999. [Viitattu: 10.5.2016]. Saatavissa: `https://www.w3.org/TR/html4/`.

[17] Luke Stevens and RJ Owen. *The Truth about HTML5*. Apress, 2013.

[18] Peter Lubbers, Brian Albers, Frank Salim, and Tony Pye. *Pro HTML5 programming*. Springer, 2011.

[19] Anthony Hickson. *HTML, The living standard*. Lulu.com, 2011.

[20] Web Hypertext Application Technology Working Group et al. Html living standard. [Viitattu: 20.5.2016]. Saatavissa: `http://www.whatwg.org/specs/web-apps/current-work/multipage/`.

[21] R Cabanier, J Mann, J Munro, T Wiltzius, and I Hickson. Html canvas 2d context, w3c recommendation rec-2dcontext-20151119/, november 2015. [Viitattu: 3.6.2016]. Saatavissa: `https://www.w3.org/TR/2dcontext/`.

[22] Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor, Silvia Pfeiffer, and Ian Hickson. Html 5.1 specification. *W3C Working Draft*, 2016. [Viitattu: 22.5.2016]. Saatavissa: `https://www.w3.org/TR/2016/WD-html51-20160310/`.

[23] Wallace Jackson. *HTML5 Quick Markup Reference*. Apress, 2016.

[24] Mark Pilgrim. *HTML5: up and running*. "O'Reilly Media, Inc.", 2010.

[25] Mikael Olsson. *JavaScript Quick Syntax Reference*. Apress, 2015.

[26] Eric A Meyer. *Cascading style sheets: The definitive guide*. "O'Reilly Media, Inc.", 2004.

[27] Charles Severance. Javascript: Designing a language in 10 days. *Computer*, (2):7–8, 2012.

[28] David Flanagan. *JavaScript: the definitive guide*. "O'Reilly Media, Inc.", 2006.

[29] Mojtaba Mehrara, Po-Chun Hsu, Mehrzad Samadi, and Scott Mahlke. Dynamic parallelization of javascript applications using an ultra-lightweight speculation mechanism. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 87–98. IEEE, 2011.

[30] Google. Chrome v8 introduction. [Viitattu: 21.6.2016]. Saatavissa: `https://developers.google.com/v8/intro`.

[31] Mozilla. Spidermonkey. [Viitattu: 21.6.2016]. Saatavissa: `https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey`.

[32] Marija Selakovic and Michael Pradel. Performance issues and optimizations in javascript: an empirical study. In *Proceedings of the 38th International Conference on Software Engineering*, pages 61–72. ACM, 2016.

[33] Ed Anuff. *The Java sourcebook: a complete guide to creating Java applets for the Web*. John Wiley & Sons, Inc., 1996.

[34] Ken Arnold, James Gosling, David Holmes, and David Holmes. *The Java programming language*, volume 2. Addison-wesley Reading, 2000.

[35] Oracle. Moving to a plugin-free web. [Viitattu: 15.5.2016]. Saatavissa: `https://blogs.oracle.com/java-platform-group/entry/moving_to_a_plugin_free`.

[36] Stephen Downes. Places to go: Youtube. *Innovate: Journal of Online Education*, 4(5):5, 2008.

[37] Steve Jobs. Thoughts on flash. *Apple, Inc*, 2010. [Viitattu: 22.3.2016]. Saatavissa: `http://www.apple.com/hotnews/thoughts-on-flash/`.

[38] Steve Fulton and Jeff Fulton. *HTML5 canvas*. "O'Reilly Media, Inc.", 2013.

[39] Maged N Kamel Boulos, Jeffrey Warren, Jianya Gong, and Peng Yue. Web gis in practice viii: Html5 and the canvas element for interactive online mapping. *International journal of health geographics*, 9(1):1, 2010.

[40] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in html5. *Proceedings of W2SP*, 2012.

[41] Chris Marrin. Webgl specification. *Khronos WebGL Working Group*, 2011. [Viitattu: 28.3.2016]. Saatavissa: `https://www.khronos.org/registry/webgl/specs/1.0.3/`.

[42] Tony Parisi. *WebGL: up and running*. "O'Reilly Media, Inc.", 2012.

[43] M Douglas McIlroy, JM Buxton, Peter Naur, and Brian Randell. Mass-produced software components. 1968.

[44] Hafedh Mili, Fatma Mili, and Ali Mili. Reusing software: Issues and research directions. *IEEE transactions on Software Engineering*, 21(6):528–562, 1995.

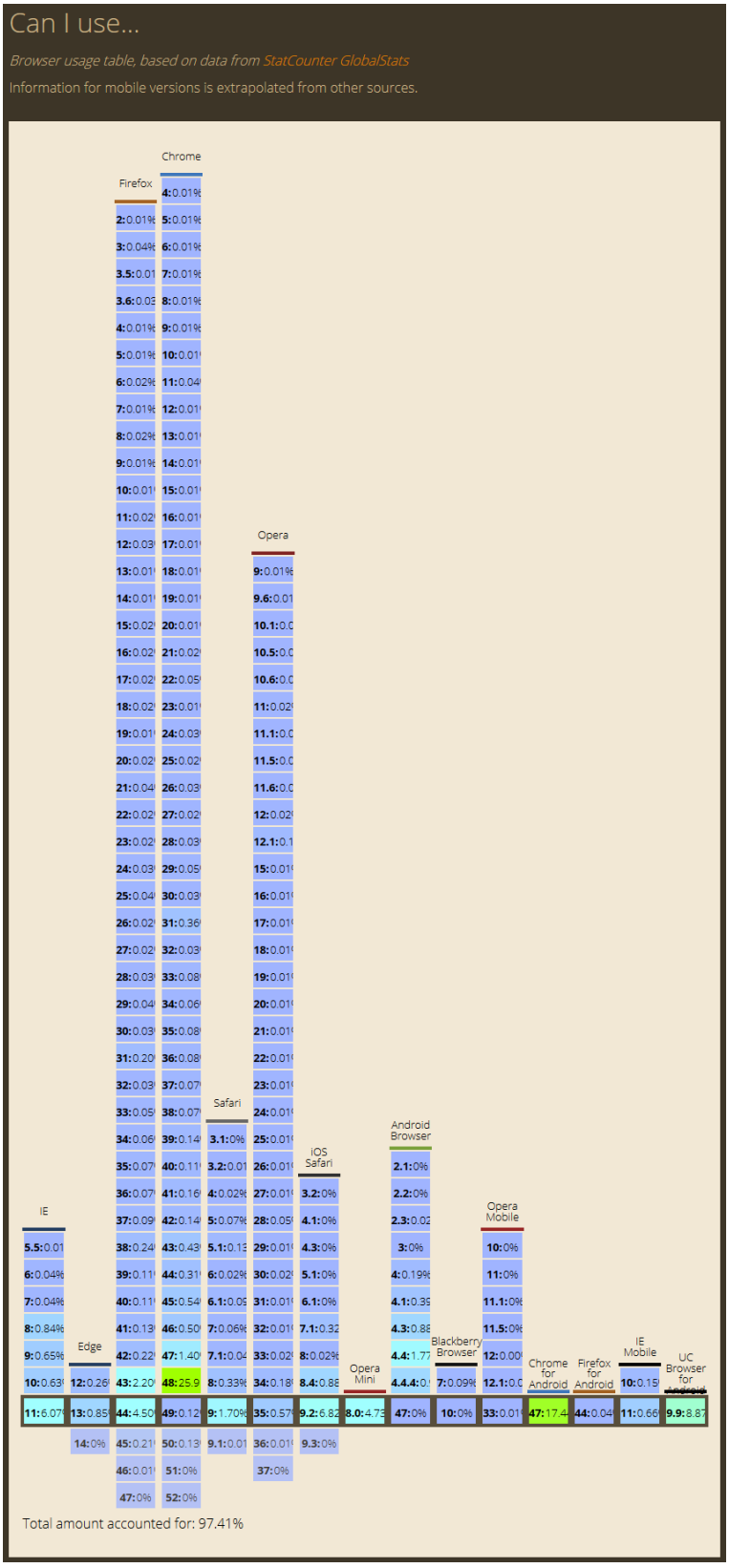
[45] Ian Sommerville. *Software Engineering. International computer science series*. 2007.

[46] Brad Green and Shyam Seshadri. *AngularJS*. "O'Reilly Media, Inc.", 2013.

[47] Ralph E Johnson. Frameworks=(components+ patterns). *Communications of the ACM*, 40(10):39–42, 1997.

Liite A

Selaintuki Canvas- ja WebGL-tekniikoille



Kuva A.1: Caniuse.com-sivuston käyttämät selainten markkinaosuudet helmikuulta 2016. Caniuse.com käyttää mobiiliselaimia lukuunottamatta StatCounter GlobalStats -sivuston http://gs.statcounter.com julkaisemia selainten markkinaosuuksia. Kuvankaappaus Caniuse.com-sivustolta 9.3.2016.

Browsers not included on site

The following tables display the browsers logged by StatCounter that are not accounted for on the caniuse.com website.

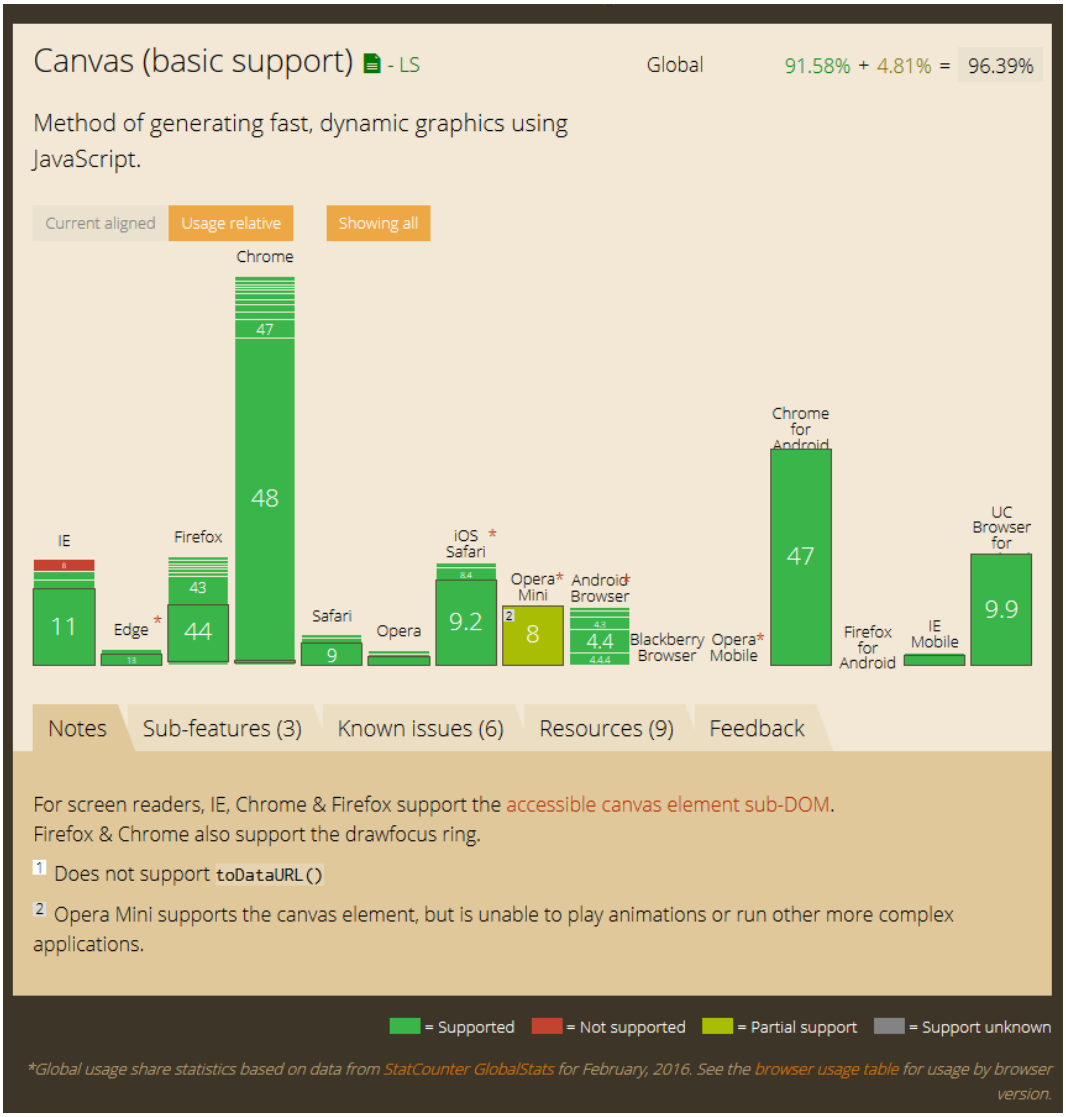
Desktop browser versions

StatCounter ID	Users
Other	0.274%
Yandex Browser 16.2	0.117%
Yandex Browser 15.12	0.117%
Maxthon 4.4	0.101%
Safari 6.2	0.073%
Sony PS4 0	0.061%
Coc Coc 53.2	0.056%
IE 0	0.040%
Chromium 25.0	0.039%
Chrome 48.2	0.039%
Phantom 0	0.034%
Chromium 48.0	0.022%
QQ Browser 9.3	0.022%
Sony PS3 0	0.017%
Chrome 47.6	0.017%
Chrome 47.1	0.017%
Chrome 48.4	0.017%
QQ Browser 9.2	0.017%
Mozilla 0	0.017%
Opera 12.5	0.017%
Amigo 45.0	0.011%
Chrome 48.5	0.011%
Chrome 46.1	0.011%
Yandex Browser 13.10	0.011%
Yandex Browser 15.9	0.011%
IE 4.0	0.006%
Chromium 47.0	0.006%
Coc Coc 52.2	0.006%
Coc Coc 50.2	0.006%
Pale Moon 26.0	0.006%
Firefox 40.1	0.006%
SeaMonkey 2.39	0.006%
Safari 2.0	0.006%
Yandex Browser 15.10	0.006%
Opera 7.1	0.006%
Yandex Browser 15.7	0.006%
NetFront NX 3.0	0.006%
Chrome 39.6	0.006%
Total:	1.241%

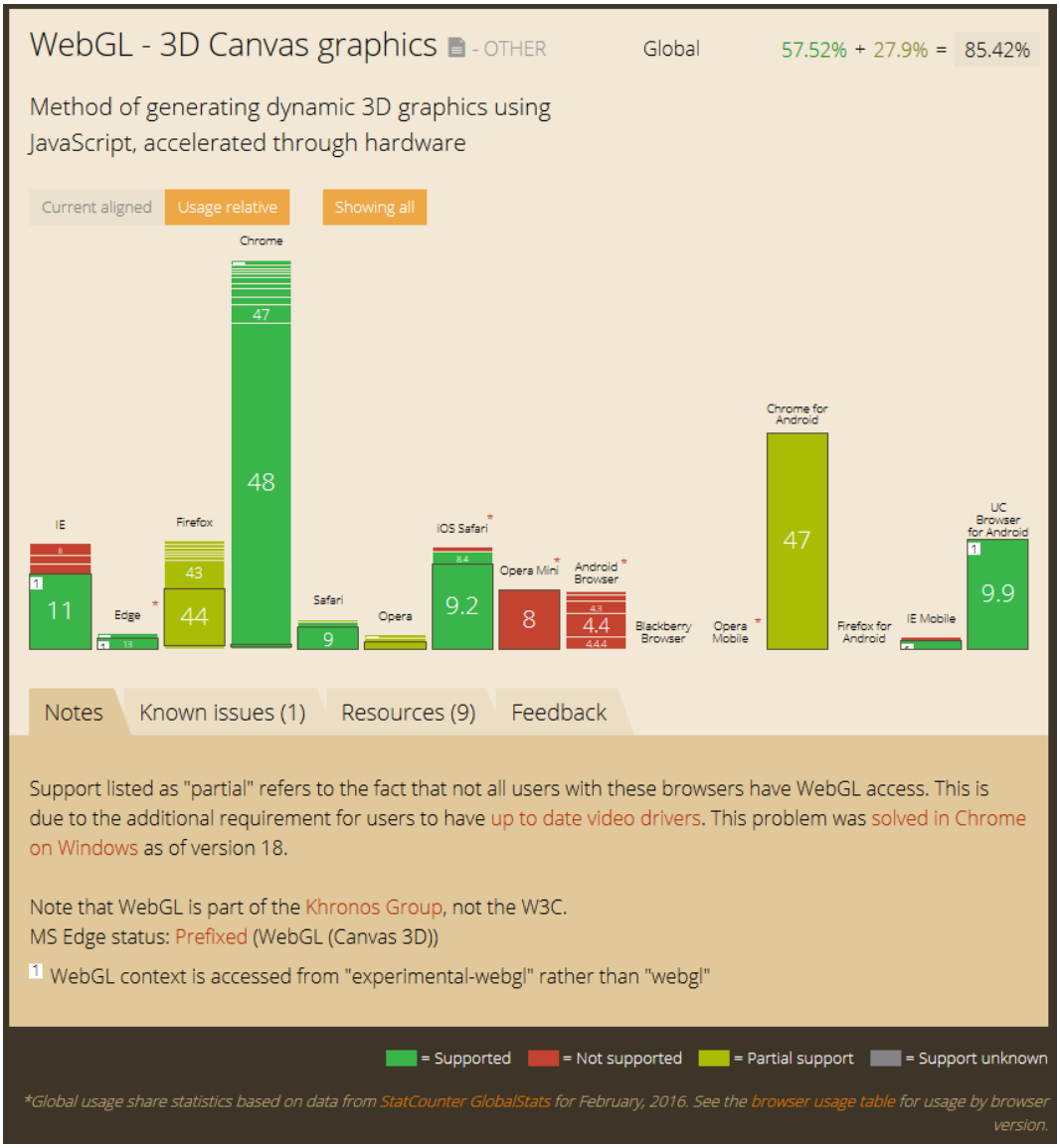
Mobile browsers

StatCounter ID	Users
Samsung Internet	0.622%
BlackBerry (pre-OS7)	0.258%
Nokia	0.150%
NetFront	0.093%
Unknown	0.084%
QQ Browser	0.040%
Puffin	0.040%
Dolfin	0.026%
Edge	0.022%
Samsung	0.018%
Tizen	0.018%
Opera Mini	0.018%
SonyEricsson	0.018%
Baidu Browser	0.018%
MeeGo	0.018%
Yandex Browser	0.018%
Other	0.013%
Sony PSP Vita	0.013%
Openwave Mobile Browser	0.013%
Phantom	0.009%
Obigo	0.004%
Total:	1.512%

Kuva A.2: StatCounter GlobalStats -sivuston http://gs.statcounter.com mittaamat selainten markkinaosuudet, joita ei ole käytetty Caniuse.com-sivuston tekniikkavertailussa. Kuvankaappaus Caniuse.com-sivustolta 9.3.2016.



Kuva A.3: Caniuse.com-sivuston yhteenveto HTML5 Canvas -tekniikkaa tukevista selaimista. Kuvankaappaus Caniuse.com-sivustolta 9.3.2016.



Kuva A.4: Caniuse.com-sivuston yhteenveto WebGL-tekniikkaa tukevista selaimista. Kuvankaappaus Caniuse.com-sivustolta 9.3.2016.